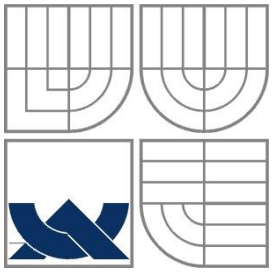
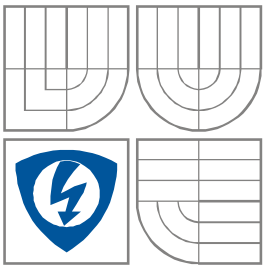


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

Softwarová knihovna pro převod a analýzu signálů

Software library for signal transform and analysis

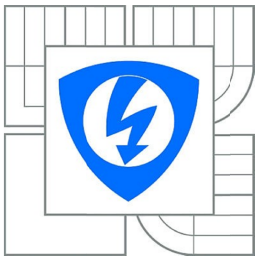
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Daniel Skopalík
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Roman Maršálek, Ph.D.
SUPERVISOR

KONZULTANT — Ing. Slavomír Skopalík
CONSULTANT

BRNO, 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Bakalářská práce

bakalářský studijní obor
Elektronika a sdělovací technika

Student:	Daniel Skopalík	***ID:***	155236
Ročník:	3	***Akademický rok:***	2014/2015

NÁZEV TÉMATU:

Softwarová knihovna pro převod a analýzu signalů

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte algoritmus pro převzorkování diskrétního signálu s neekvidistantním vzorkováním na signál s ekvidistantním vzorkováním, který co nejvhodněji doplní (interpoluje) vzorky, které byly zahozeny filtrem pro odstranění redundantních dat. Uvažte náročnost algoritmu na systémové zdroje a rychlost výpočtu. Navržený algoritmus naprogramujte v jazyce C#. Vytvořte novou knihovnu do které bude umístěna funkce implementující výše popsaný algoritmus.

Do vámi vytvořené knihovny doplňte další funkce pro výpočet vzájemné korelace mezi dvěma signály a funkce pro výpočet autokorelace, frekvenčního spektra, numerické integrace a derivace pro jeden signál. Srovnejte vámi implementované funkce s jinými dostupnými funkcemi. Cílová platforma je Microsoft Windows.

DOPORUČENÁ LITERATURA:

[1] C# Language reference [Online]. 2014 - [cit. 12. června 2014]. Dostupné na: http://msdn.microsoft.com/en-us/library/618ayhy6.aspx.
[2] NEVŘIVA, P. Analýza signálů a soustav. 1. vyd. Praha: BEN - technická literatura, 2000, 671 s. ISBN 80-730-0004-0.

Termín zadání:	9.2.2015	***Termín odevzdání:***	28.5.2015

Vedoucí práce: doc. Ing. Roman Maršálek, Ph.D.
Konzultanti bakalářské práce:

doc. Ing. Tomáš Kratochvíl, Ph.D.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Firma Elekt Labs s.r.o. se zabývá sběrem a vyhodnocováním dat převážně z výrobních strojů. Pro vyhodnocení a prezentaci dat využívá vlastní informační systém MASA (Měřící a Statistická Aplikace). Cílem této práce je vyvinout softwarové funkce, které uživatelům přinesou nové možnosti při analýze dat. Nejdůležitějšími novými funkcemi bude zobrazení derivace či integrace průběhu časově proměnného signálu. Dalšími funkcemi bude převzorkování signálu, Fourierova analýza, korelace signálů a statistická korelace.

Klíčová slova

Převzorkování signálu, Derivace signálu, Integrace signálu, korelace signálů, Rychlá Fourierova transformace.

Abstract

Firm Elekt Labs s.r.o. is focused on data collection and analysis from machines usually in production industry. Firm use their own information system MASA (Measuring and Statistic Application) for data presentation and analysis. Goal of this bachelor thesis is implement new software functions which bring new features in data analysis. Most important new functions will be show derivation or integration of time variable signal. Also functions for signal oversampling, Fourier analysis, signal correlation and statistic correlation will be implemented.

Keywords

Signal oversampling, Non equidistant to equidistant transform, signal derivation, signal integration, Signal correlation, fast Fourier transform.

SKOPALÍK, D. *Softwarová knihovna pro převod a analýzu signálů*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Ústav radioelektroniky, 2015. 44 s. Bakalářská práce. Vedoucí práce: doc. Ing. Roman Maršálek, Ph.D.

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma Softwarová knihovna pro převod a analýzu signálů jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

V Brně dne ....

(podpis autora)

Poděkování

Děkuji vedoucímu bakalářské práce doc. Ing. Romanu Maršálkovi, Ph.D. a Ing. Slavomíru Skopalíkovi za účinnou metodickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne ....

(podpis autora)

Obsah

Abstrakt ... 3
Klíčová slova ... 3
Abstract ... 3
Keywords ... 3
Prohlášení ... 5
Poděkování ... 5
Seznam obrázků ... 8
1 Úvod ... 9
2 Analýza současného stavu a implementace nové funkcionality ... 10
2.1 Popis současného stavu systému ... 10
2.1.1 Měření stavu strojů ... 10
2.1.2 Sběr naměřených dat ... 10
2.1.3 Filtr ... 10
2.1.4 Vzorkovač ... 11
2.1.5 Úložiště dat ... 12
2.1.6 Prezentace dat ... 12
2.2 Popis nově implementované funkcionality a jejího využití v praxi ... 13
2.2.1 Rekonstrukce průběhu signálu na základ uložených vzorků ... 13
2.2.2 Numerická integrace ... 14
2.2.3 Numerická derivace ... 14
2.2.4 Frekvenční spektrum ... 14
2.2.5 Korelace ... 14
3 Teoretický rozbor řešení ... 15
3.1 Rekonstrukce průběhu signálu ... 15
3.1.1 Schodovitá interpolace ... 16
3.1.2 Lineární interpolace ... 17
3.1.3 Lagrangeova interpolace ... 18
3.2 Numerická integrace ... 20
3.2.1 Lichoběžníková metoda ... 20
3.2.2 Obdélníková metoda ... 20
3.3 Numerická derivace ... 22
3.3.1 Robustnější metody pro výpočet derivace ... 22
3.4 Korelace ... 23
3.4.1 Korelační koeficient ... 23

3.4.2 Korelační funkce ... 23
3.5 Konvoluce ... 25
3.6 Fourierova transformace ... 26
3.6.1 Algoritmu radix 2 DIT ... 26
3.7 Přesnost výpočtů v oboru reálných čísel na PC ... 28
4 Programovací jazyk C# ... 29
4.1 Knihovny ... 29
4.2 Pokročilejší programové konstrukce využité v této práci ... 29
4.2.1 Abstraktní třída ... 29
4.2.2 Statická funkce ... 30
4.2.3 Vkládání funkcí (Inlining) ... 30
4.2.4 Kritická sekce ... 30
5 Realizace ... 31
5.1 Srovnání interpolačních funkcí pomocí Matlab ... 31
5.2 Knihovna ElSignalTransform ... 33
5.2.1 Struktura Sample ... 33
5.2.2 Třída Signal ... 33
5.2.3 Třída Interpolation ... 34
5.2.4 Rozhraní iPolynomialInterpolator ... 34
5.2.5 Třída LinearInterpolator ... 34
5.2.6 Třída LagrangeInterpolator ... 34
5.2.7 Třída Integration ... 35
5.2.8 Třída Derivation ... 35
5.2.9 Třída Statistics ... 36
5.2.10 Třída FFT ... 36
5.2.11 Třída CrossCorelation ... 37
5.2.12 Třída Convolution ... 37
5.2.13 Třída Summator ... 37
5.3 Ukázková aplikace ... 39
6 Závěr ... 41
6.1 Dosažené výsledky ... 41
6.2 Srovnání s jinými softwarovými knihovnami ... 42
6.3 Plány pro budoucí rozvoj ... 43
7 Použité zdroje informací ... 44

Seznam obrázků

Obrázek 1 - Blokové schéma sběru dat 10
Obrázek 2 - Vývojový diagram funkce vzorkovače 11
Obrázek 3 - Ukázka chyby způsobené nevhodnou lineární interpolací 13
Obrázek 4 - Graf znázorňující tolerančí pole rekonstruované funkce 15
Obrázek 5 - Doplnění vzorků a interpolace po částech 16
Obrázek 6 - Ukázka Schodovité interpolace 17
Obrázek 7 - Ukázka lineární interpolace 18
Obrázek 8 - Ukázka Lagrangeovy interpolace 19
Obrázek 9 - Ukázka integrace lichoběžníkovou metodou 20
Obrázek 10 - Srovnání integrace dle pravidla pravého a levého koncového bodu 21
Obrázek 11 - Grafická ukázka vzájemně korelační funkce 24
Obrázek 12 - Grafická ukázka autokorelační funkce 24
Obrázek 13 - Grafická ukázka konvoluční funkce 25
Obrázek 14 - Diagram datových toků algoritmu DIT. Podle [6] 27
Obrázek 15 - Znázornění použití algoritmu DIT pro posloupnost N=8. Převzato z [6] 27
Obrázek 16 - Graf testovací funkce a různých interpolací 31
Obrázek 17 - Graf kvadratických odchylek různých interpolací 32
Obrázek 18 - prezentační aplikace - záložka importu dat. 39
Obrázek 19 - prezentační aplikace - záložka grafu. 39
Obrázek 20 - prezentační aplikace - zobrazení spektra signálu 40
Obrázek 21 - Rozdělení signálu do spojitých části a interpolace po částech 41

1 Úvod

Informační systém MASA (Měřící a statistická aplikace) od firmy Elekt Labs s.r.o. mimo jiných funkcí které jsou obvyklé pro výrobní informační systémy, umožňuje sběr dat o stavu strojů a zařízení, uložení dat, a zobrazení uživatelům pomocí webového rozhraní.

Cílem této práce je vytvořit funkce pro širší možnosti analýzy průběhu signálu. Webové rozhraní v současné době umožňuje pouze zobrazení dat v tabulce, případně v grafu. Nové funkce by měly uživatelům informačního systému přinést přehlednější pohled na naměřená data a také zviditelnit souvislosti v datech, které dosud nebyly zřejmé. Mezi nové funkce patří:

- Numerická derivace v čase – umožní například z odečtů stavu elektroměru vypočítat průběh spotřeby v čase.
- Numerická integrace v čase – umožní například z dat o aktuálním průtoku kapaliny vypočítat její spotřebované množství.
- Výpočet vzájemné korelační funkce a autokorelační funkce
- Výpočet korelačního koeficientu
- Výpočet konvoluce dvou signálů
- Výpočet frekvenčního spektra

Pro některé z těchto funkcí je nezbytné, aby vstupní data byla tvořena ekvidistantními vzorky. Z tohoto důvodu bude součástí nové knihovny také funkce pro rekonstrukci signálu, která zajistí převod vstupních dat, která jsou kvůli nižším nárokům na úložný prostor tvořena neekvidistantními vzorky, na data tvořena ekvidistantními vzorky které co nejlépe odpovídají měřeným datům, která byla vzorkována.

Z licenčních důvodů budou nové softwarové funkce umístěny do samostatně šiřitelné knihovny, jejíž zdrojové kódy jsou součástí přílohy bakalářské práce. Pro účely demonstrace funkčnosti je vytvořena jednoduchá aplikace, která dokáže přehledně v grafech zobrazit výstupy funkcí implementovaných v knihovně.

2 Analýza současného stavu a implementace nové funkcionality

2.1 Popis současného stavu systému

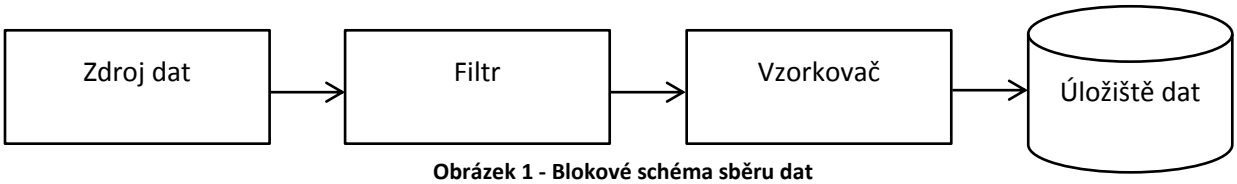
2.1.1 Měření stavu strojů

Hodnoty měřených veličin, jakými jsou například teplota, tlak, průtok, rychlost, jsou čteny z PLC (Programovatelný logický automat), které jsou umístěny v rozvaděčích strojů a jsou k nim připojeny snímače měřených veličin.

PLC jsou dále připojeny k aplikačnímu serveru, obvykle sítí Ethernet případně jiným způsobem, který je vhodný v průmyslovém prostředí. Například sériovou linkou RS485.

2.1.2 Sběr naměřených dat

Sběr dat je prováděn serverovou aplikací, která čte naměřená data z PLC, případně jiných zdrojů a provádí jejich filtraci, vzorkování a uložení do databáze. Parametry vzorkování jsou pro každou měřenou veličinu individuálně nastaveny tak, aby byla s dostatečnou přesností zachycena dynamika změn měřených parametrů, a zároveň aby uložená data zabrala co nejméně místa na pevném disku.



Obrázek 1 - Blokové schéma sběru dat

2.1.3 Filtr

Hlavním účelem tohoto filtru je redukce šumu a zlepšení přesnosti měření. Tento filtr realizuje funkci klouzavého průměru z posledních N_f hodnot.

$$y = \frac{\sum_{i=0}^{N} x_{[i]}}{N} \text{ kde } N_f \in \langle 1, \left\lceil \frac{T_{Oversampling}}{T_{Sampling}} \right\rceil \rangle$$

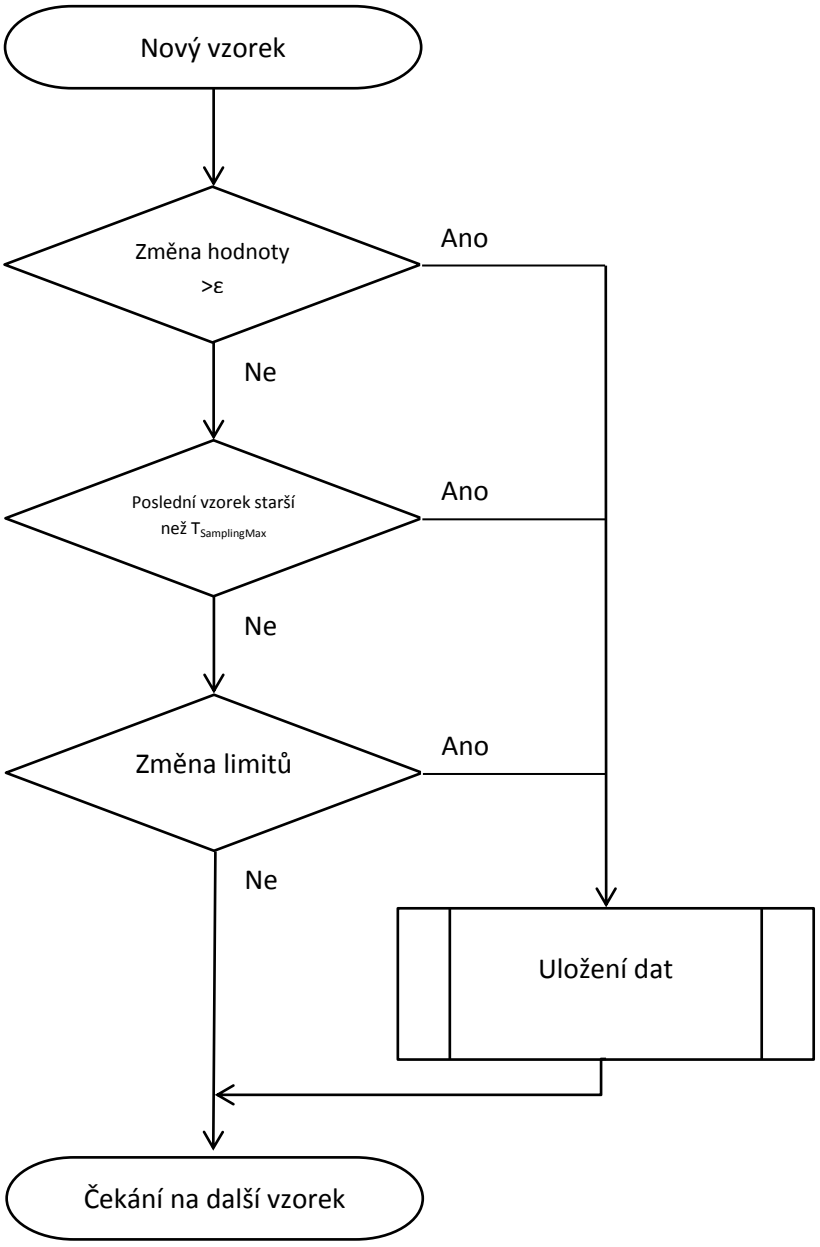
Kde N_f je proměnný počet vzorků ve frontě, které splňují podmínku, že čas uplynulý od jejich změření je menší než doba $T_{Oversampling}$ (Čas převzorkování). V případě že vzorkovač detekuje změnu, která bude uložena do databáze, je také vyprázdněna fronta filtru, aby byly lépe zachyceny rychlé změny.

Perioda čtení dat ze zdroje (dále $T_{Sampling}$)je u většiny sledovaných signálů nastavena na 1s. čas $T_{Oversampling}$ je nastaven na 10s.

2.1.4 Vzorkovač

Vzorkovač pro každý vstupní vzorek vypočítá rozdíl oproti poslednímu uloženému vzorku. Pokud tento rozdíl přesáhne nastavené ε - konstanta definující minimální velikost změny, která bude uložena, bude aktuální vzorek i s časovou značkou bude uložen do databáze.

Uložení vzorku může být vynuceno také z důvodu překročení maximální nastavené doby od uložení posledního vzorku ($T_{SamplingMax}$) případně při změně limitů či žádané hodnoty. Celý proces znázorňuje následující vývojový diagram:



Obrázek 2 - Vývojový diagram funkce vzorkovače

2.1.5 Úložiště dat

Pro uložení naměřených dat je využívána open source SQL databáze Firebird. Do databáze je ukládána naměřená hodnota, čas a datum měření a další informace jako je důvod uložení hodnoty (například pravidelné měření nebo měření vynuceno vnějším zásahem).

V databázi jsou dále pro každý měřený signál uloženy metadata jako název, jednotky ve kterých je veličina měřena a popis. V dalších tabulkách jsou uloženy limity pro dané měření (minimální a maximální hodnoty měřené veličiny, jejichž překročení může signalizovat poruchu) a historie změn těchto limitů.

2.1.6 Prezentace dat

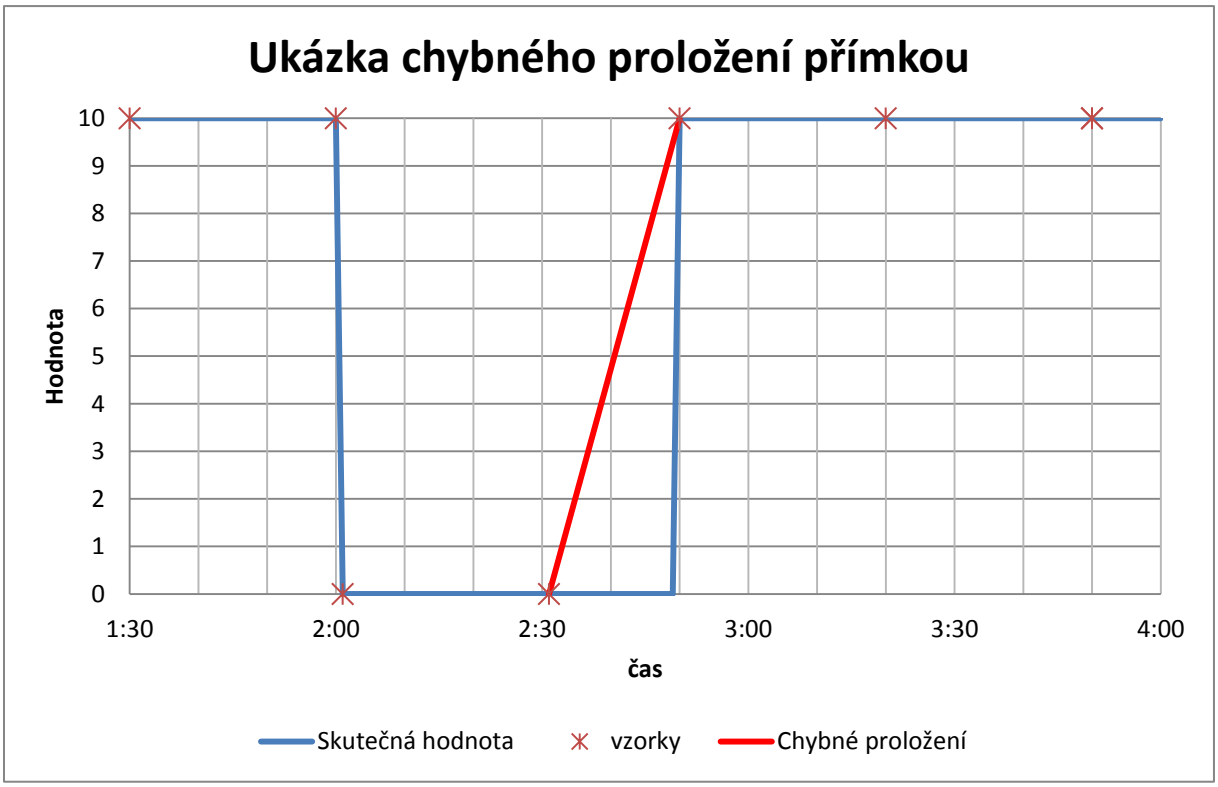
Data jsou prezentována pomocí webového rozhraní informačního systému MASA. V modulu diagnostických signálů je možné vybrat jeden či více signálů a pro zvolené časové období načíst data z databáze. Data mohou být zobrazena v tabulce nebo může být vygenerován graf. Grafy jsou generovány pomocí knihovny ZedGraph. Průběh v grafu může být zobrazen pomocí schodovité interpolace nebo propojením dvou sousedních bodů přímkou, která je realizována již v knihovně ZedGraph.

2.2 Popis nově implementované funkcionality a jejího využití v praxi

2.2.1 Rekonstrukce průběhu signálu na základ uložených vzorků

Ukládání vzorků pouze v případě, kdy dochází ke změnám hodnoty měřeného parametru, má nízké nároky na úložný prostor. Ovšem použití těchto dat je pro některá vyhodnocení obtížné či dokonce nemožné. Rekonstrukce průběhu je nezbytná především pro další analyzování průběhu, případně pro zobrazení průběhu v grafu, který bude více podobný reálnému signálu.

Častým jevem v měřených datech jsou skokové změny měřené veličiny ve chvílích, kdy došlo například k zapnutí a vypnutí stroje. Pokud je tento jev zachycen malým počtem vzorků, které od sebe jsou časově značně vzdáleny, protože většina z nich byla uložena jako pravidelné měření v čase, může dojít k situaci, jaká je zobrazena na následujícím grafu. Pokud by z těchto vzorků byla vypočítána derivace v čase, tak by derivace v intervalu 2:30 až 2:50 nebyla závislá na původním průběhu signálu, ale na délce intervalu mezi posledním uloženým vzorkem a skokovou změnou hodnoty měřené veličiny.



Obrázek 3 - Ukázka chyby způsobené nevhodnou lineární interpolací

Tato chyba by způsobila značnou nepřesnost ve všech výsledcích, které jsou závislé na tvaru signálu. Například derivace, integrace či frekvenční analýza. Výpočet korelace nad těmito průběhy je téměř nemožný, protože by mohlo docházet ke skrytí podobnosti signálů.

Tuto chybu by bylo možné odstranit použitím schodovité interpolace, ovšem do signálu by byly zaneseny nové problémy, především obohacení spektra o vyšší frekvenční složky, které v původním signálu nebyly přítomny.

Funkce pro rekonstrukci průběhu se bude skládat ze dvou hlavních částí:

1. Detekce nespojitých změn a doplnění chybějících vzorků
2. Proložení vzorků vhodnou funkcí tak, aby bylo možno provést převzorkování dle požadavků.

Výstupem této funkce bude signál, který se bude svým průběhem co nejvíce podobat původnímu vzorkovanému průběhu.

2.2.2 Numerická integrace

Integraci lze využít u případů, kdy je měřen okamžitý stav, jako je například průtok chemikálie, nebo rychlost odvíjení materiálu z role a chceme znát kumulovanou hodnotu za čas. Například spotřebovaný objem chemikálie nebo metráž vstupního materiálu.

2.2.3 Numerická derivace

Derivaci lze využít naopak v případech, kdy je měřen kumulativní stav, jako je například stav elektroměru, a chceme znát průběh změn v čase tedy například odběr elektřiny v čase.

2.2.4 Frekvenční spektrum

Frekvenční spektrum může pomoci například při hledání oscilací regulované soustavy.

2.2.5 Korelace

Identifikace specifických průběhů ve sledovaném signálu. Využití například pro detekci cyklů stroje.

2.2.5.1 Korelační koeficient

Vhodné pro kontrolu závislosti kvalitativního parametru na nastavení stroje, případně určení míry lineární závislosti dvou sledovaných veličin.

2.2.5.2 Autokorelace

Identifikace periodicky se opakujících úseků signálu. Například vyhledání specifických částí stavů stroje jako jsou pracovní cykly, nebo poruchové stavy.

2.2.5.3 Korelace dvou signálů

Identifikace souvislostí mezi signály, například posouzení vlivu rychlosti výroby stroje na spotřebu energií.

3 Teoretický rozbor řešení

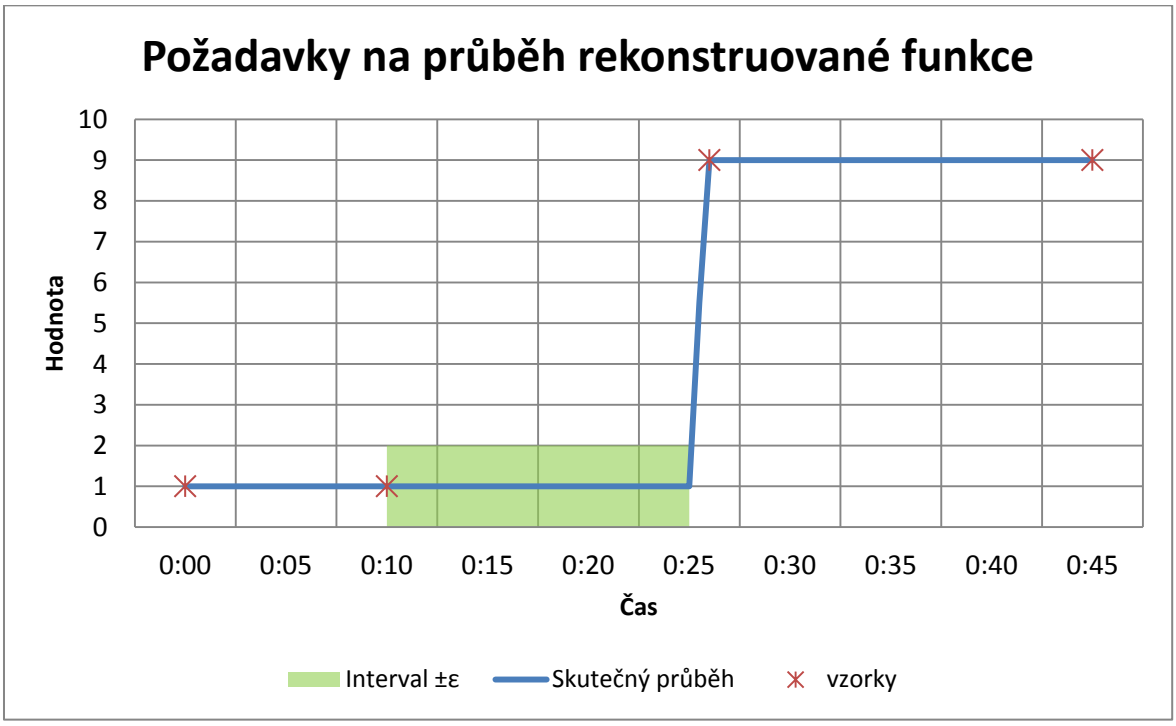
3.1 Rekonstrukce průběhu signálu

Hlavní motivací pro rekonstrukci původního průběhu signálu je převod z neekvidistantního vzorkování na ekvidistantní. Při tomto procesu je potřebné odhadnout hodnotu funkce v časových okamžicích, které odpovídají novému ekvidistantnímu vzorkování. Proto je potřebné z uložených vzorků dat obnovit původní tvar signálu, který byl vzorkován a vhodnou interpolační funkcí a hodnoty této interpolační funkce navzorkovat v potřebných časových okamžicích.

Aby bylo možné z uložených dat co nejlépe rekonstruovat původní průběh signálu, je nezbytné data zpracovat algoritmem, jehož funkce bude inverzní k funkci vzorkovače popsaného v předchozí kapitole. Pro správnou funkčnost je nezbytné, aby bylo splněno několik vstupních požadavků:

- V okamžiku spuštění rekonstrukčního algoritmu jsou známy parametry nastavení vzorkovače.
- Budeme předpokládat, že v čase kdy byla změřena vstupní data, nedošlo k výpadku měřícího systému na dobu kratší než je nastavená doba $T_{SamplingMax.}$ Takovýto výpadek nelze z naměřených vzorků zjistit.
- Budeme předpokládat, že v čase kdy byla změřena vstupní data, nastavení systému bylo takové, že byl splněn Nyquistův vzorkovací teorém, a při vzorkování signálu nedošlo k aliasingu ve frekvenční oblasti.

Pokud jsou tyto podmínky splněny, je možné dále předpokládat, že v čase mezi vzorky x_i a x_{i+1} hodnota signálu byla v intervalu $< x_i - \varepsilon; x_i + \varepsilon >$.

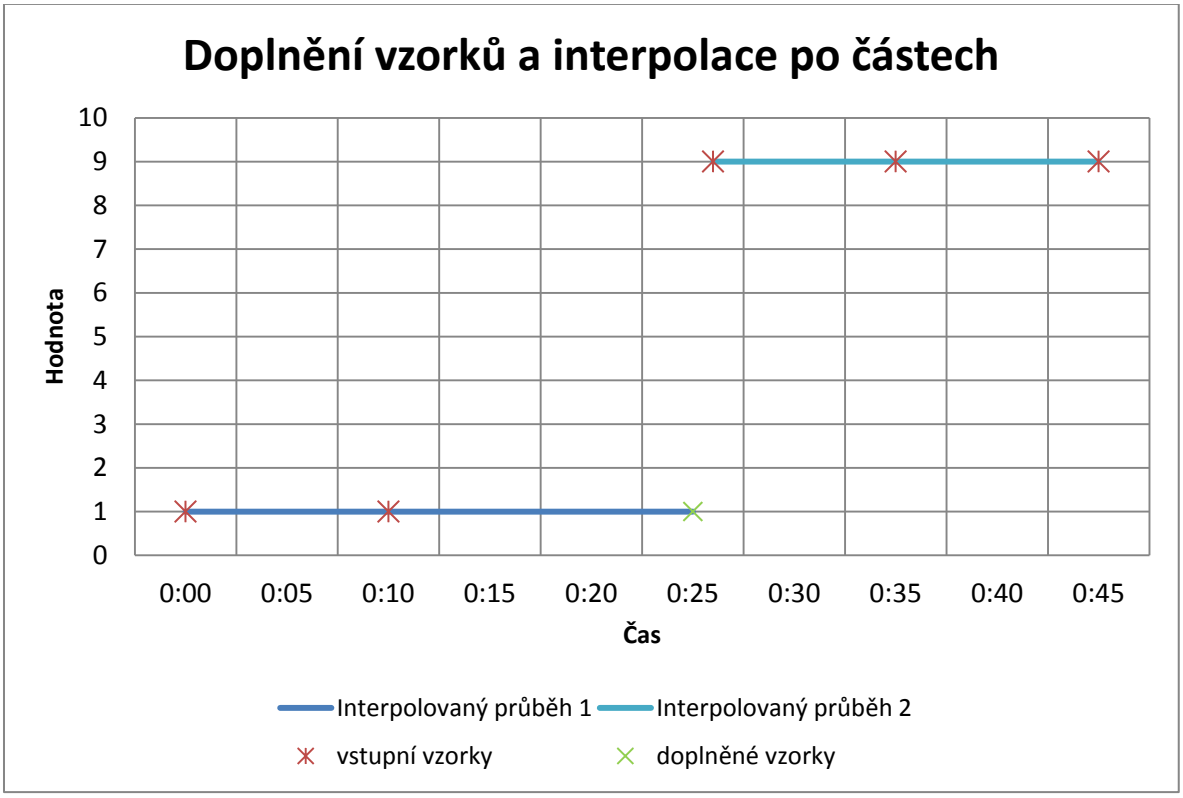


Obrázek 4 - Graf znázorňující toleranční pole rekonstruované funkce

Při analýze uložených vzorků lze předpokládat, že pokud je časový rozdíl mezi dvěma po sobě následujícími vzorky větší než $T_{SamplingMin,}$ a zároveň absolutní velikost rozdílu hodnot těchto vzorků je větší než ε, došlo k velmi rychlé změně měřené veličiny. V případě, že dojde k této rychlé změně, níže popisované interpolační algoritmy s výjimkou schodovité interpolace, by nedokázaly správně napodobit původní průběh funkce, a došlo by k vychýlení z požadovaného intervalu $< x_i - \varepsilon; x_i + \varepsilon >$.

Předpokládejme, že vzorek x_i je posledním uloženým vzorkem před rychlou změnou a vzorek x_{i+1} je prvním vzorkem po rychlé změně. Aby bylo možné správně odhadnout původní průběh pomocí interpolační funkce, musí tato funkce splňovat podmínku, že její hodnota v bodě, který je na časové ose vzdálen o $T_{SamplingMin}$ od bodu x_{i+1} doleva, musí být hodnota v intervalu $< x_i - \varepsilon; x_i + \varepsilon >$.

Jedním ze způsobů, jak výše popsané podmínky splnit, je k měřeným vzorkům doplnit nový vzorek. Interpolační funkce pak budou procházet tímto bodem. Protože jsou interpolační funkce funkcemi spojitými, je nezbytné vstupní vzorky rozdělit do několika částí, na kterých je možné průběh funkce interpolovat pomocí takovýchto funkcí. Popsaná situace je znázorněna v následujícím grafu.



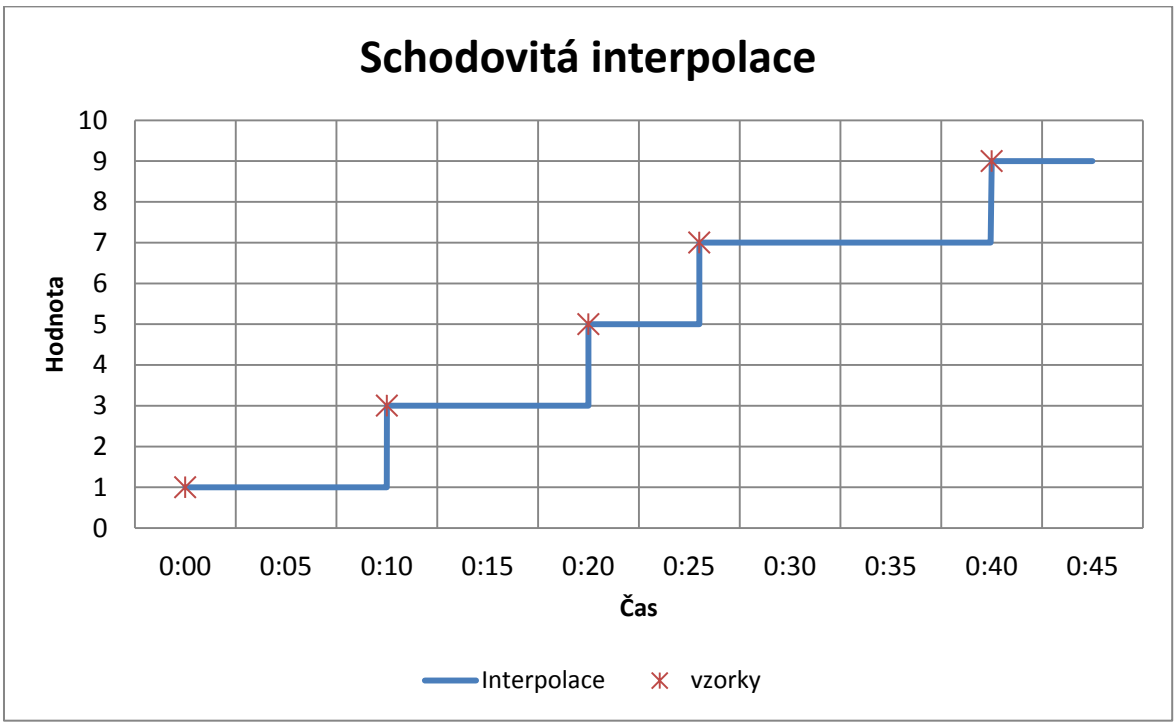
Obrázek 5 - Doplnění vzorků a interpolace po částech

3.1.1 Schodovitá interpolace

Schodovitá interpolace je nejjednodušší interpolační funkcí. Interpolační funkce je po částech spojitou funkcí. Hodnota této funkce se mění v bodech se známou hodnotou (uzlových bodech) a je konstantní na intervalu mezi dvěma sousedními body.

Pokud známe uzlové body o souřadnicích $[x_0,y_0]$ a $[x_1,y_1]$ tak pro interpolační funkci platí:

$$y = y_0 \text{ pro } x \in < x_0, x_1)$$



Obrázek 6 - Ukázka Schodovité interpolace

Výhody schodovité interpolace:

- Velmi nízké výpočetní nároky.
- Robustnost - nemohou vzniknout nežádoucí oscilace či extrémy.

Nevýhody

- Velké odchylky od interpolované funkce.

3.1.2 Lineární interpolace

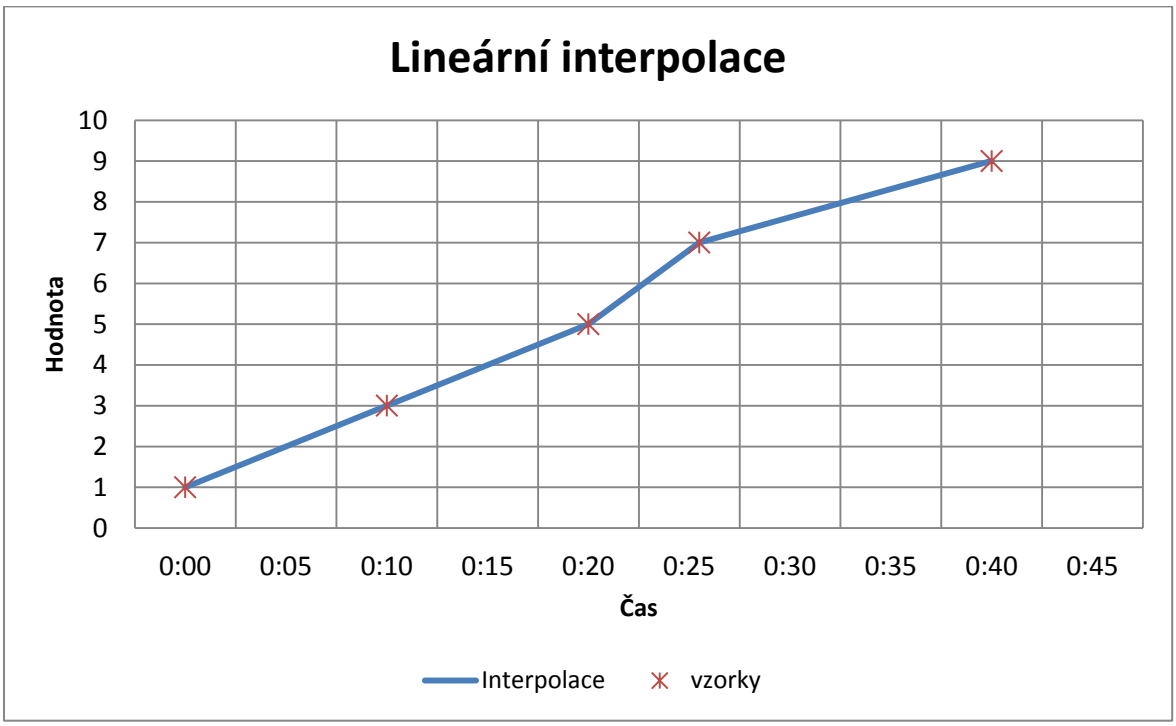
Lineární interpolace je metoda, při které je interpolovaná funkce v intervalu mezi uzlovými body proložena přímkou. Interpolační funkce (přímka) je zadána rovnicí

$$y(x) = ax + b$$

Koeficienty a a b je možné vypočítat, pokud známe 2 body, kterými přímka prochází. Pokud známe body o souřadnicích $[x_0,y_0]$ a $[x_1,y_1]$ je možné koeficienty vypočítat pomocí následujících vztahů[1]:

$$a = \frac{y_0 - y_1}{x_0 - x_1} \qquad b = y_0 - a * x_0$$

Pro interpolaci na intervalu $<x_0,x_1>$ jsou koeficienty konstantní a stačí je vypočítat pouze jednou.



Obrázek 7 - Ukázka lineární interpolace

Výhody lineární interpolace:

- Nízké výpočetní nároky.
- Robustnost - nemohou vzniknout nežádoucí oscilace či extrémy.

Nevýhody

- Nižší přesnost především při nízkém počtu vzorků.

3.1.3 Lagrangeova interpolace

Lagrangeova interpolace spočívá v proložení N+1 známých bodů (uzlů interpolace) pomocí Lagrangeova polynomu N stupně.[1]

Vstupními daty pro konstrukci Lagrangeova polynomu jsou body o souřadnicích $[x_0,y_0]$ až $[x_n,y_n]$.

$$L_n(x) = f(x_0)l_0(x) + \cdots + f(x_n)l_n(x)$$

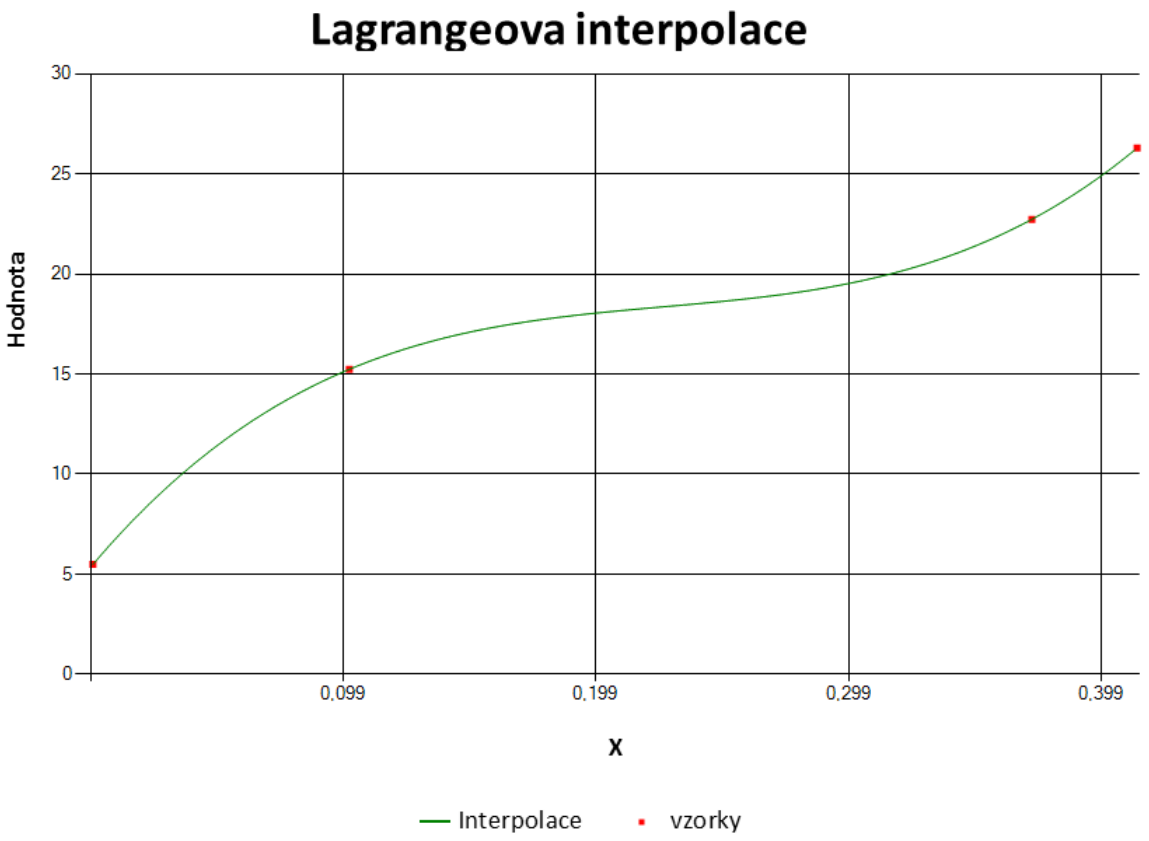
Kde pro $l_i(x)$ platí:

$$l_i(x_j) = \begin{cases} 1 \text{ pro } i \neq j \\ 0 \text{ pro } i = j \end{cases}$$

Těmto podmínkám vyhovuje polynom

$$l_i(x) = \prod_{0 \leq m \leq n} \frac{x - x_m}{x_i - x_m}$$

Pro počítačové řešení Lagrangeova polynomu lze využít výhody, že dělitel výše uvedeného výrazu není závislý na x, a celou část dělitele stačí vypočítat jednou pro interpolaci na intervalu x_0 až x_n.



Obrázek 8 - Ukázka Lagrangeovy interpolace

Výhody Lagrangeovy interpolace:

- Dokáže dobře interpolovat goniometrické či exponenciální funkce i při nízkém počtu uzlových bodů

Nevýhody

- Může docházet k nežádoucím oscilacím, pokud dojde k rychlé změně v průběhu aproximované funkce, případně v její první derivaci.

3.2 Numerická integrace

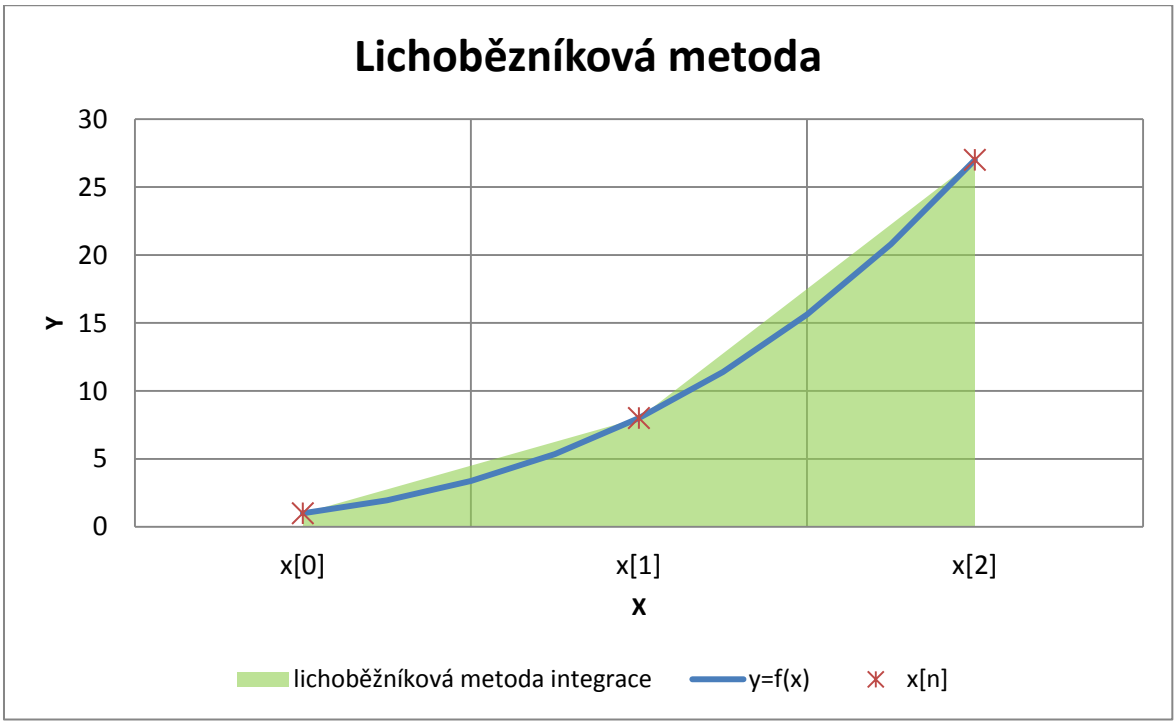
Naměřená data, jsou uložena jako jednotlivé vzorky v čase. Jejich popsání pomocí primitivní matematické funkce by bylo velmi náročné, v některých případech prakticky nemožné. Z tohoto důvodu pro výpočet integrálu nelze použít analytických metod a hodnota integrálu bude počítána pouze přibližně, pomocí numerických metod. [1]

Pro výpočet integrálu ze vstupních dat bude použito složené lichoběžníkové metody, případně složené obdélníkové metody.

Pokud máme průběh $y = f(x)$ zaznamenán pomocí N diskrétních bodů $[x_0, y_0]$ $až$ $[x_n, y_n]$ kde $y_n = f(x_n)$, potom je výpočet integrálu rozdělen do N částí, ve kterých lze provést výpočet integrálu $\int_{x_n}^{x_{n+1}} f(x)dx$ některou z jednoduchých metod numerické integrace. Výpočet integrálu za celý úsek definovaný $[x_0, y_0]$ $až$ $[x_n, y_n]$ je sumou integrálů v jednotlivých částech.

3.2.1 Lichoběžníková metoda

Lichoběžníková metoda, v literatuře někdy nazývaná jako lichoběžníkové pravidlo, je velmi jednoduchým způsobem pro přibližný výpočet hodnoty integrálu. Při volbě vhodně krátkého intervalu integrace poskytuje uspokojivě přesné výsledky.



Obrázek 9 - Ukázka integrace lichoběžníkovou metodou

Hodnota integrálu v intervalu x_n až x_{n+1} je vypočítána pomocí následujícího vzorce [2]:

$$\int_{x_n}^{x_{n+1}} f(x) \doteq (x_{n+1} - x_n) * \frac{f(x_n) + f(x_{n+1})}{2}$$

3.2.2 Obdélníková metoda

Obdélníková metoda v literatuře někdy nazývaná jako obdélníkové pravidlo je výkonově méně náročná na výpočet než lichoběžníková metoda. U specifických průběhů vstupní funkce může

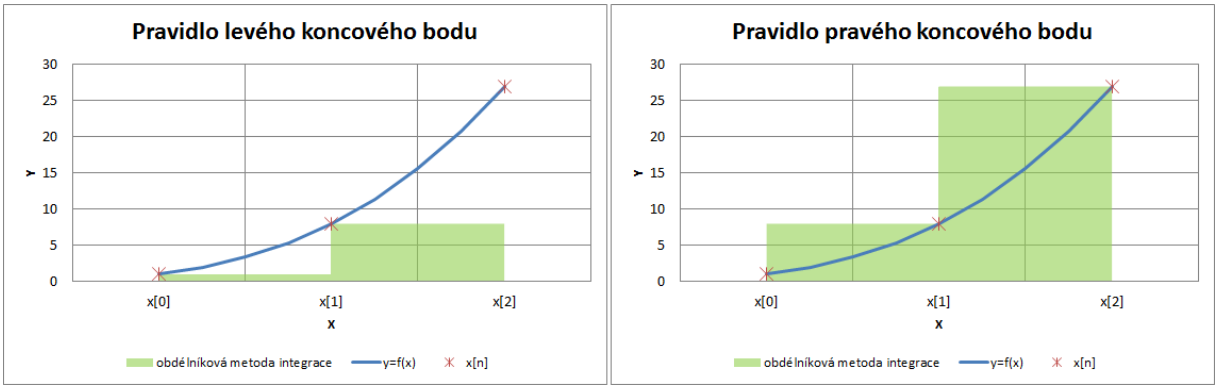
poskytnout přesnější výsledek výpočtu integrálu. Obdélníková metoda má 2 možné varianty: pravidlo levého koncového bodu a pravidlo pravého koncového bodu. [2]

Při použití varianty pravidlo levého koncového bodu je hodnota integrálu v intervalu x_n až x_{n+1} je vypočítána pomocí vzorce.

$$\int_{x_n}^{x_{n+1}} f(x) \doteq f(x_{\boldsymbol{n}})(x_{n+1} - x_n)$$

Varianta pravého koncového bodu využívá k výpočtu výšky obdélníku pravý koncový bod intervalu.

$$\int_{x_n}^{x_{n+1}} f(x) \doteq f(x_{\boldsymbol{n+1}})(x_{n+1} - x_n)$$



Obrázek 10 - Srovnání integrace dle pravidla pravého a levého koncového bodu

3.3 Numerická derivace

Derivace je definována jako směrnice tečny k průběhu funkce v daném bodě. V praxi hodnota derivace funkce podává informaci o strmosti jejího nárůstu nebo poklesu. Stejně jako u výpočtu integrálu nebylo možné použít analytické řešení i pro výpočet derivace je nezbytné využít odhadu derivace pomocí numerických metod. U numerických metod výpočtu derivací je třeba brát v úvahu konečnou přesnost výpočtů s reálnými čísly ve výpočetní technice.

V jednoduchých numerických metodách pro odhad derivace je výpočet směrnice tečny nahrazen výpočtem směrnice sečny. Rozdíl lze při vhodné vzdálenosti bodů zanedbat [1].

$$f'(x) = \frac{f(x+h) - f(x)}{h}$$

Velikost parametru *h*, který je roven vzdálenosti vstupních bodů, má velký vliv na odchylku výsledku. Pokud je parametr h příliš velký, přímka procházející body $[x, f(x)]$ *a* $[x+h, f(x+h)]$ se značně odchyluje od původního průběhu funkce. Pokud máme průběh $y = f(x)$ zaznamenán pomocí N diskrétních bodů $[x_0, y_0]$ *až* $[x_n, y_n]$ kde $y_n = f(x_n)$ potom parametr h je roven vzdálenosti sousedních vzorků $h = x_{n+1} - x_n$. Hodnotu derivace lze poté přibližně určit pomocí vzorce:

$$y'(n) = \frac{y_{n+1} - y_n}{x_{n+1} - x_n}$$

3.3.1 Robustnější metody pro výpočet derivace

V případě že jsou vstupní data zatížena šumem, mohlo by dojít k tomu, že trend nárůstu bude v absolutní hodnotě srovnatelný, nebo menší, než amplituda šumu, a z výsledku derivace zatížené šumem by nebyl trend nárůstu nebo poklesu patrný. Proto je v těchto případech vhodné použít některý z robustnějších způsobů výpočtu derivace. [3] Například podle vzorce:

$$y'(n) = \frac{-y_{n-1} + y_{n+1}}{2\Delta x}$$

Nebo s účinnější filtrací šumu:

$$y'(n) = \frac{+y_{n-2} - 8y_{n-1} + 8y_{n+1} - y_{n+2}}{12\Delta x}$$

Kde $\Delta x = x_{n+1} - x_n$. Podmínkou pro použití těchto metod je ekvidistantní vzorkování signálu. Tyto metody výpočtu podávají výsledky, jako by před výpočtem derivace byla data filtrována FIR (filtr s konečnou impulzní odezvou), filtrem s frekvenční charakteristikou typu dolní propust pro potlačení rušení s vyšší frekvencí.

3.4 Korelace

Korelační analýzu lze rozdělit do dvou základních oblastí, jsou to statické a dynamické parametry a korelační. Statické korelační parametry vyjadřují trvalou závislost mezi sledovanými parametry. Nejdůležitějším z nich je korelační koeficient, který vyjadřuje míru lineární závislosti sledovaných veličin. Dynamické parametry korelační analýzy jsou závislé na vzájemném posunutí signálů. Mezi dynamické parametry korelační analýzy patří vzájemně korelační funkce a autokorelační funkce. Tyto korelační funkce vyjadřují závislost míry závislosti signálů na jejich vzájemném posunutí. **Chyba! Nenalezen zdroj odkazů.**

3.4.1 Korelační koeficient

Existuje mnoho druhů korelačních koeficientů, nejběžněji používaný je párový korelační koeficient v literatuře často nazýván Pearsonův korelační koeficient, který vyjadřuje míru těsnosti lineární stochastické vazby mezi sledovanými veličinami. [5]

Jiné korelační koeficienty mohou poskytnout například robustnější výsledky při výskytu odlehlých hodnot. Jejich využití však vyžaduje pokročilé znalosti statistiky a dobré znalosti procesů, ze kterých vycházejí naměřená data.

Párový korelační koeficient nabývá hodnot z intervalu -1 až 1. Kladná hodnota koeficientu vyjadřuje přímou úměru mezi analyzovanými veličinami, záporná hodnota nepřímou úměru. Pokud je korelační koeficient r = 0 je to znakem, že mezi sledovanými veličinami neexistuje lineární závislost. Naopak pokud se hodnota korelačního koeficientu blíží jedné z limitních hodnot, naznačuje to lineární závislost.

Párový korelační koeficient lze vypočítat pomocí následujícího vzorce [5]:

$$r_{x,y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} = \frac{cov(x,y)}{\sigma_x \sigma_y}$$

Kovarianci lze vypočítat pomocí následujícího vzorce [15]:

$$cov(x,y) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{N}$$

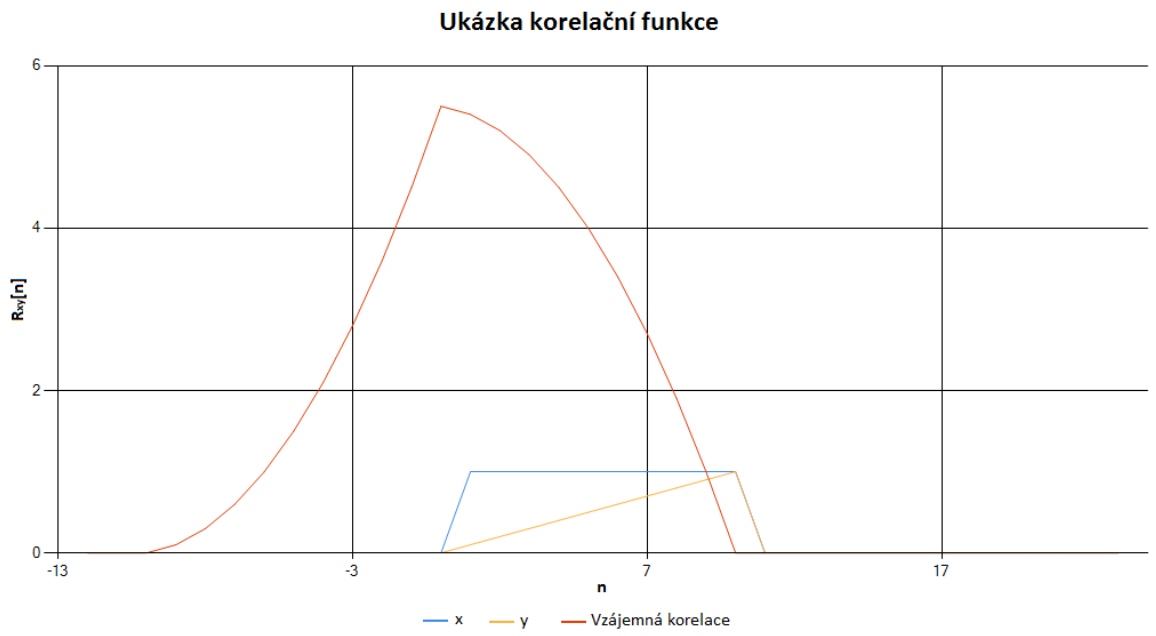
3.4.2 Korelační funkce

Vzájemná korelační funkce poskytuje informaci o podobnosti tvaru analyzovaných signálů v závislosti na jejich vzájemném posunutí. Autokorelační funkce poskytuje informace periodicitě signálu. V oboru signálů se spojitým časem je korelace definována pomocí následujícího vzorce [4]:

$$R_{x,y}(\tau) = \int_{-\infty}^{\infty} x(t)y(t-\tau)\,dt$$

Analogicky pro signály s diskrétním časem je korelační funkce definována následovně [4]:

$$R_{x,y}[n] = \sum_{m=1}^{N} x_m y_{m-n}$$



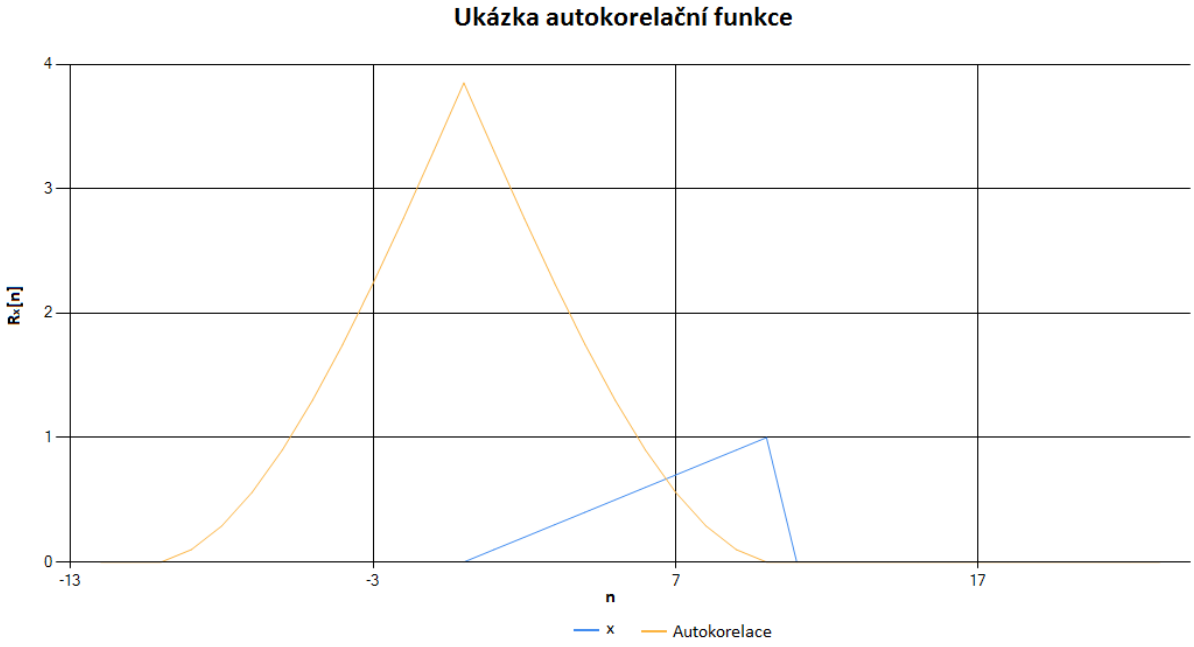
Obrázek 11 - Grafická ukázka vzájemně korelační funkce

Autokorelační funkce je vzájemnou korelační funkcí, u níž jsou dva vstupní signály reprezentovány jedním. Vyjadřuje podobnost tvaru funkce sobě samotné, v závislosti na vzájemném posunutí. U signálů se spojitým časem může být vyjádřena vzorcem:

$$R_x(\tau) = \int_{-\infty}^{\infty} x(t)x(t-\tau)\,dt$$

Analogicky pro signál s diskrétním časem:

$$R_x[n] = \sum_{m=1}^{N} x_m x_{m-n}$$



Obrázek 12 - Grafická ukázka autokorelační funkce

3.5 Konvoluce

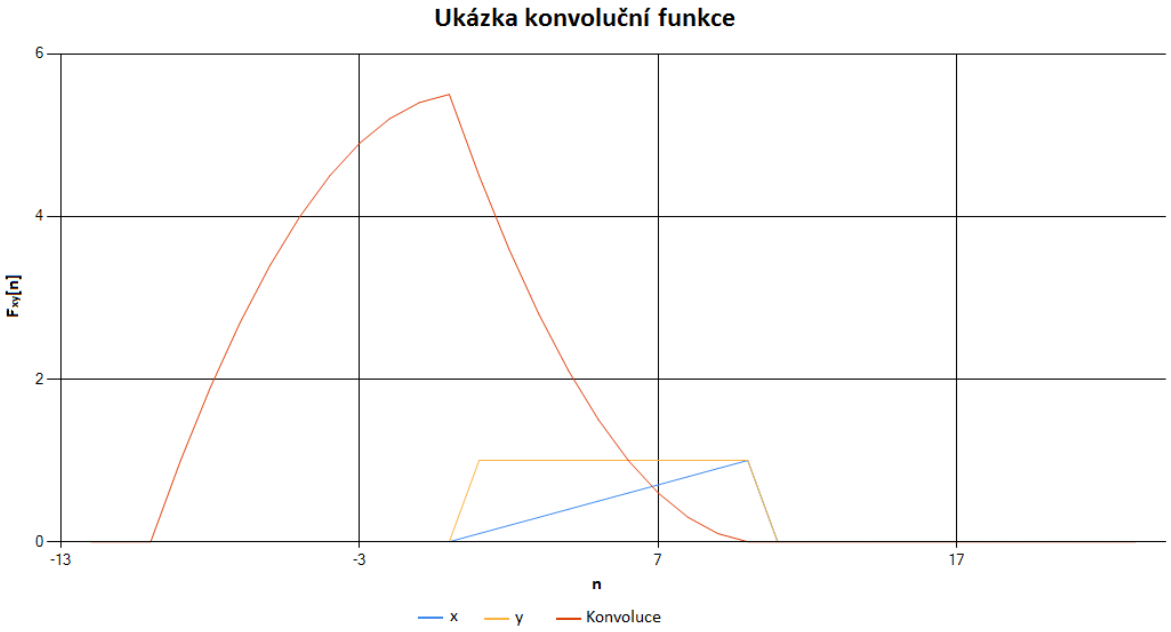
Konvoluce je matematický operátor zpracovávající dvě vstupní funkce, jehož výstupem je třetí funkce. Využití nalézá při analýze a modelování lineárních systému či soustav kde platí, že výstup soustavy lze modelovat jako konvoluci vstupního signálu a impulsní odezvy soustavy. Výpočet konvoluce je velmi podobný výpočtu vzájemné korelace signálů s tím, že jeden ze vstupních signálů je časově překlopen (zrcadlen kolem osy Y).

Pro signály se spojitým časem je toto vyjádřeno vzorcem [4]:

$$S_{x,y}(\tau) = \int_{-\infty}^{\infty} x(t)y(\tau - t)\,dt$$

Analogicky pro signály s diskrétním časem je korelační funkce definována následovně:

$$S_{x,y}[n] = \sum_{m=1}^{N} x_m y_{n-m}$$



Obrázek 13 - Grafická ukázka konvoluční funkce

3.6 Fourierova transformace

Fourierova transformace je transformací, která převádí funkci či signál z časové oblasti do frekvenční oblasti. Výstup Fourierovy transformace nese informaci o zastoupení jednotlivých spektrálních složek v analyzovaném signálu. Diskrétní signály lze z posloupnosti vzorků v časové oblasti do frekvenční oblasti převést pomocí DFT (diskrétní Fourierovy transformace). Z důvodu velké časové náročnosti výpočtu DFT zvláště u dlouhých vstupních posloupností, se v praxi využívají algoritmy pro výpočet rychlé Fourierovy transformace. Nejčastěji používaným algoritmem pro výpočet FFT je Cooley-Turkey radix 2 DIT (Decimation In Time) algoritmus. **Chyba! Nenalezen zdroj odkazů.**

Transformaci DFT lze popsat následující rovnicí:

$$X(k/NT) = T\sum_{n=0}^{N-1} x(NT)e^{-j2\pi kn/N}$$

Kde N je počet vzorků vstupní posloupnosti a T je perioda vzorkování. Výpočetní náročnost tohoto algoritmu je $O(N^2)$. Pro lepší čitelnost následujících rovnic je v literatuře často zaváděn "twiddle factor" $W_N = e^{-j2\pi/N}$. DFT lze nyní popsat rovnicí:

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{nk}$$

3.6.1 Algoritmu radix 2 DIT

Hlavní myšlenkou zrychlení výpočtu pomocí FFT algoritmu je rozdělení výpočtu do dvou posloupností o délce N/2. Pokud je délka vstupní posloupnosti $N = 2^k$ kde k je libovolné celé číslo, lze rekurzivně rozdělovat výpočet do dvou částí až do dosažení výpočtu 2-bodové DFT. Původně kvadratická složitost výpočtu se díky tomu sníží na $O(N \log_2(N))$. [6]

Vstupní posloupnost *x* o délce N je rozdělena do dvou částí x_1 a x_2 o délce N/2 na sudé a liché členy. Potom lze stejného výsledku dosáhnout pomocí:

$$X[k] = \sum_{n=0}^{N/2-1} x_1[n]W_{N/2}^{nk} + W_N^k \sum_{n=0}^{N/2-1} x_2[n]W_{N/2}^{nk}$$

Což lze upravit do tvaru:

$$X[k] = X_1[k] + W_N^k X_2[k]$$

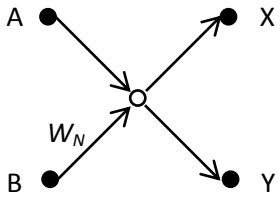
Výše uvedená rovnice definuje výpočet první poloviny vzorků. Důvodem této skutečnosti je, že *k* nabývá hodnot pouze z intervalu <0; N/2 - 1>. Pro výpočet druhé poloviny vzorků lze využít toho že:

$$W_N^{k+N/2} = -W_N^k$$

A hodnoty druhé poloviny vzorků určit pomocí vzorce:

$$X[k + N/2] = X_1[k] - W_N^k X_2[k]$$

Algoritmus radix 2 DIT bývá díky znázornění diagramu datových toků často nazýván jako motýlek

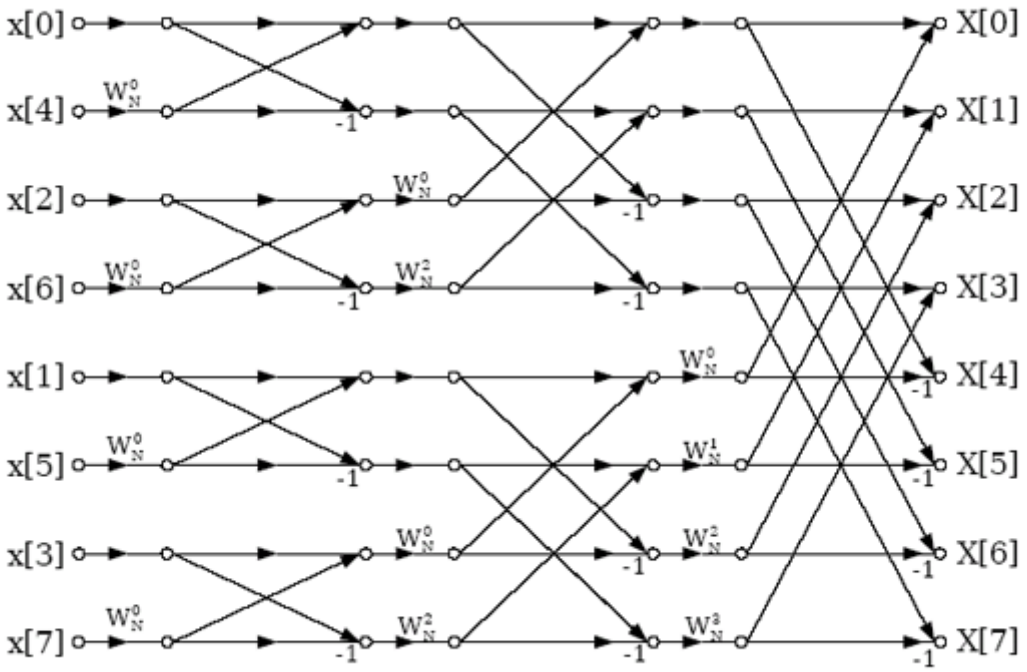


Obrázek 14 - Diagram datových toků algoritmu DIT. Podle [6]

Vyjádřeno pomocí vzorce:

$$X = A + B\ W_N^k \quad Y = A - B\ W_N^k$$

Využití algoritmu radix 2 DIT pro výpočet FFT pro delší (8mi bodovou) vstupní posloupnost je znázorněno na následujícím obrázku:



Obrázek 15 - Znázornění použití algoritmu DIT pro posloupnost N=8. Převzato z [6]

Před využitím radix 2 DIT je nezbytné vstupní posloupnost přeuspořádat do tzv. bitově reverzního pořadí [12].

3.7 Přesnost výpočtů v oboru reálných čísel na PC

Pro vyjádření reálných čísel se v informatice obvykle používá formát s plovoucí řádovou čárkou dle normy IEEE 754 [13]. Norma IEEE 754 definuje několik možných reprezentací čísel s pohyblivou řádovou čárkou, které se lišší svou přesností. Běžně používanými formáty jsou:

- binary32 - Sigle precision (základní přesnost) – číslo vyjádřeno pomocí 32 bitů, z toho 23bitů mantisa, 8 bitů exponent a 1bit znaménko.
- Binary64 - Double precision (dvojitá přesnost) - číslo vyjádřeno pomocí 64 bitů, z toho 52bitů mantisa, 11 bitů exponent a 1bit znaménko.

Norma definuje také formáty extended x86 (rozšířená dvojitá přesnost) a binary128 (čtyřnásobná přesnost) Tyto formáty mají sice výrazně vyšší přesnost nicméně jejich použití na PC je komplikované. Běžné procesory PC architektury x86 či x86-64 nemají hardware podporu pro výpočty s reálnými čísly ve čtyřnásobné přesnosti. Použití této přesnosti by bylo velice náročné na dobu výpočtu. Formát extended x86 je hardwarem běžných počítačů podporován ale jeho využití v .NET framework je komplikované a také by znesnadnilo přenos knihovny na jinou hardware platformu než x86.

Nejvýrazněji by se mohly numerické chyby projevit při sčítání čísel, která od sebe dělí mnoho řádů. V takovém případě by došlo ke ztrátě přesnosti menšího z přičítaných čísel, případně by se výsledek po přičtení velmi malého čísla vůbec nezměnil. Tato situace může nastat například u numerického výpočtu hodnoty integrálu z velkého množství vstupních dat. Numerické chyby, vzniklé při sčítaní čísel rozdílné velikosti, lze minimalizovat použitím metody kompenzovaného součtu.[9] Pro potřeby přesného sčítání u výpočtů integrálů byl v rámci této bakalářské práce implementován Kahanův sumační algoritmus [14].

4 Programovací jazyk C#

Programovací jazyk C# je vysokoúrovňový, objektově orientovaný programovací jazyk od firmy Microsoft. Syntaxe tohoto programovacího jazyka je podobná C++. Oproti C++ ale přináší některé značné výhody. Těmi pro programátory nejvýznamnějšími jsou:

- Automatický garbage collector (nepoužívané objekty jsou z paměti automaticky odstraněny)
- Hlídaní hranice pole – pokus o přístup k indexu pole mimo jeho aktuální rozsah je ukončen výjimkou.
- Hlídání použití neinicializovaných objektů – pokus o použití neinicializovaného objektu je ukončen výjimkou
- Velice kvalitní dokumentace – k dispozici je přehledná dokumentace i zdrojové kódy systémových funkcí.

Jednou z nevýhod jazyka C#, respektive všech jazyků platformy .NET, je nižší výkon ve srovnání s C++. Program napsaný v jazyce C# není při kompilaci převeden do zdrojového kódu cílové platformy tak, jak je to běžné například u C++ ale je zkompilován do jazyka MSIL (Microsoft Intermediate Language). MSIL je objektově orientovaný programovací jazyk, který není běžně lidmi editován. Jeho cílem je umožnit běh zkompilovaného programu na různých hardwarových architekturách. Spuštění programu je možné díky CLI (Common language interface), který vytváří abstraktní vrstvu nad fyzickým procesorem počítače, a za běhu převádí instrukce MSIL jazyka do strojových instrukcí procesoru. [10] CLI je na platformě Microsoft Windows tvořeno Microsoft .NET frameworkem, na ostatních platformách jako je Linux či Unix jsou funkce CLI zabezpečeny pomocí Mono.

4.1 Knihovny

Knihovnou je v informatice obvykle označován balík, který obsahuje funkce, definice tříd, či rozhraní a další data. Hlavním důvodem pro použití knihoven je možnost sdílet stejné funkce mezi několika programy. Knihovna není samostatně funkčním celkem, bývá referencována jiným programem, který využívá funkce této knihovny. V následující kapitole bude popsán vývoj dynamicky linkované knihovny. Výsledkem bude soubor s příponou .dll, který budou využívat jiné programy pro provádění funkcí v knihovně.

4.2 Pokročilejší programové konstrukce využité v této práci

V této části bude stručně popsán význam a využití některých pokročilejších programových konstrukcí, které by nemusely být známé čtenáři se základními znalostmi v oblasti objektového programování.

4.2.1 Abstraktní třída

Abstraktní třída je obvykle využívána jako společný předek pro několik jiných tříd. Není možné vytvořit instanci (objekt) abstraktní třídy. Potomkem abstraktní třídy již obvykle bývá běžná třída, jejíž instanci je možno vytvořit. Tato třída obsahuje všechny proměnné a musí obsahovat deklarace všech funkcí, které jsou definovány v rodičovské abstraktní třídě.

4.2.2 Statická funkce

Funkce třídy, která může být zavolána, aniž by bylo nutné vytvořit instanci této třídy. Obvykle jsou všechna potřebná data předávána pomocí parametrů.

4.2.3 Vkládání funkcí (Inlining)

Tato optimalizační technika může přinést zvýšení výkonu u velmi často volaných funkcí tím, že volání funkce nahradí kódem volané funkce

4.2.4 Kritická sekce

Kritická sekce je taková část programu, kde obvykle dochází k přístupu ke sdíleným datům a není přípustné, aby tato kritická sekce programu byla vykonávána nad stejnými daty souběžně dvěma nebo více procesy. Vstup programu do kritické sekce musí být ošetřen některým ze synchronizačních primitiv (např. zámek, semafor, …)tak, aby byl zajištěn exkluzivní přístup.

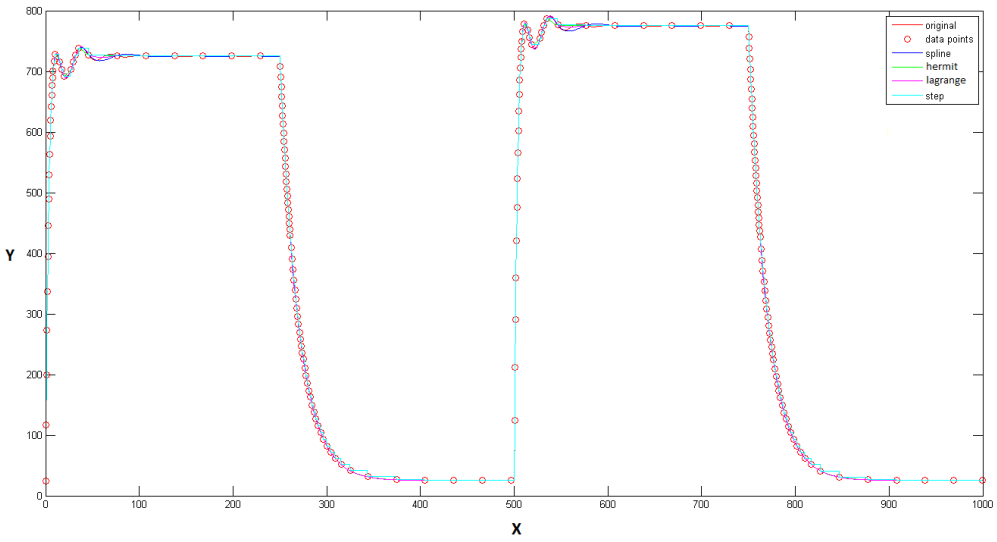
5 Realizace

5.1 Srovnání interpolačních funkcí pomocí Matlab

Před samotnou realizací interpolačních funkcí v programovacím jazyce C# byly různé interpolační funkce otestovány v prostředí Matlab. Bylo otestováno srovnání schodovité interpolace, interpolace pomocí Lagrangeova polynomu 3. řádu, Hermitovi kubické interpolace a kubického splajnu. Testy probíhaly na dvou různých testovacích funkcích.

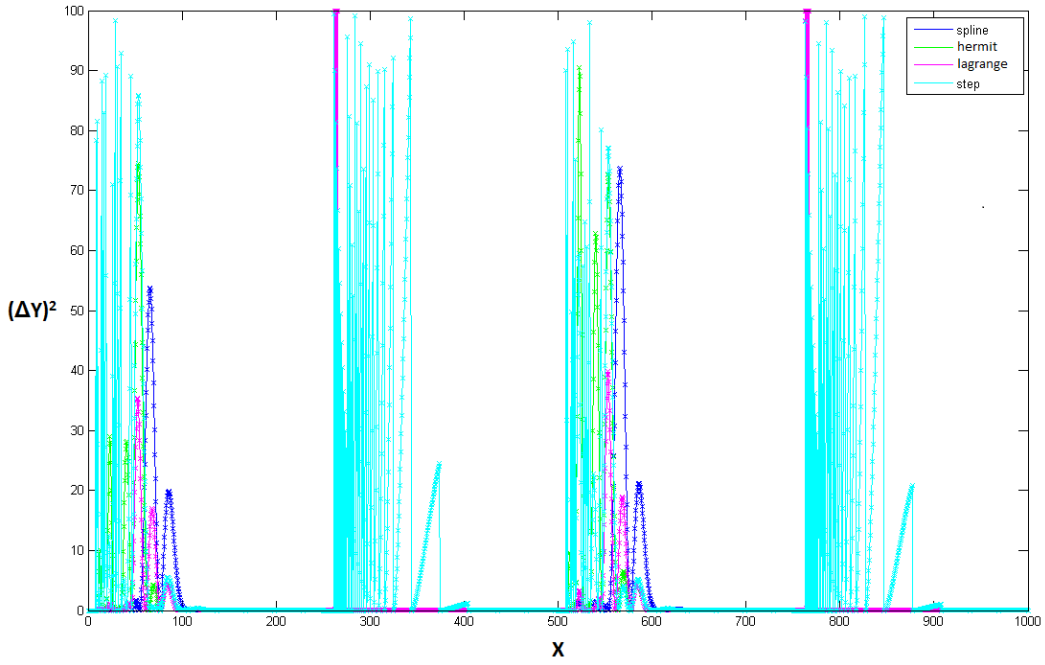
Průběh funkce byl navzorkován algoritmem, který vykonával stejnou funkci jako vzorkovač popsaný v 2. kapitole. Následně byl proveden výpočet interpolace a porovnán rozdíl mezi originální funkcí a interpolovanou funkcí. Hodnotícím kritériem byla suma kvadratických odchylek mezi interpolovanou funkcí a originální funkcí.

Testovací funkce byla definována po částech a obsahovala exponenciální části průběhu a k nim superponované sinusové rušení. Graf testovací funkce s vyznačenými vzorky (uzlovými body pro interpolaci) různými interpolacemi:



Obrázek 16 - Graf testovací funkce a různých interpolací

Druhá mocnina odchylky interpolačních funkcí od testovací funkce ve všech interpolovaných bodech je na následujícím grafu.



Obrázek 17 - Graf kvadratických odchylek různých interpolací

Na testovacích datech se ukázalo, že interpolace pomocí Lagrangeova polynomu 3. Řádu a kubického splajnu dosahují srovnatelných výsledků ve schopnostech napodobit tvar originální funkce a sumy kvadratických odchylek. Interpolace pomocí Hermitovi kubiky dobře interpoluje části, kdy konstantní hodnota přechází do růstu či poklesu a nevytváří v těchto místech nežádoucí oscilace, ovšem selhává u exponenciálních či sinusových průběhů, které jsou definovány nízkým počtem uzlových bodů. Pro další testování byla zvolena interpolace pomocí Lagrangeova polynomu.

5.2 Knihovna ElSignalTransform

Následující kapitola obsahuje stručný popis tříd a jejich funkcí dostupných v knihovně ElSignalTransform. Podrobný popis všech funkcí a jejich parametrů je obsažen v souboru nápovědy, který je součástí přílohy.

5.2.1 Struktura Sample

Je nositelem informace o jednom vzorku signálu, obsahuje dvě proměnné pojmenované X a Y. Obě jsou datového typu double.

Double je datový typ s plovoucí řádovou čárkou, který odpovídá specifikaci dle normy IEEE 754. Jedna proměnná datového typu double dle IEEE 754 zabere v pracovní paměti 64 bitů, z toho je 52 bitů vyhrazeno pro mantisu čísla, 11 bitů pro exponent a 1 bit pro znaménko. Datový typ double byl zvolen jako kompromis mezi přesností výpočtu a výpočetními nároky. Pro většinu aplikací poskytuje dostačující přesnost.

5.2.2 Třída Signal

Třída Signal nese kromě řady vzorků signálu také informace o jeho metadatech. Vzorky signálu jsou vyjádřeny pomocí pole objektů typu Sample. Mezi metadata patří informace o vzorkování signálu a jeho název.

Instanci třídy signal lze vytvořit pomocí konstruktoru bez parametrů, nebo konstruktorem s parametrem předpokládaného počtu vzorků. Pokud je při vytváření instance předem znám počet vzorků, je vhodné využít druhého konstruktoru z důvodu, že interní kolekce vzorků je již vytvořena s kapacitou odpovídající počtu vzorků. Přínosem této operace je, že potřebná paměť je alokována najednou a kolekce prvků není při postupném přidávání prvků dynamicky zvětšována.

Funkce SubSignal: Funkce je svým účelem při práci se signálem velice podobná funkci SubString při práci s textovými řetězci. Ze vstupního signálu vybere vzorky, jejichž hodnota X je v intervalu definovaném vstupními parametry start a end. Tyto vybrané vzorky jsou zkopírovány do nového signálu, do kterého jsou zkopírována také metadata zdrojového signálu.

Funkce Abs: Funkce zkopíruje všechny vzorky vstupního signálu do výstupního, včetně metadat. Y hodnota vzorku je při kopírování nahrazena absolutní hodnotou Y.

Funkce Scale: Funkce zkopíruje všechny vzorky vstupního signálu do výstupního včetně metadat. Y hodnota vzorku je při kopírování vynásobena hodnotou zadanou v proměnné scaleFactor.

Funkce Offset: Funkce zkopíruje všechny vzorky vstupního signálu do výstupního včetně metadat. Y hodnota vzorku je při kopírování sečtena s hodnotou zadanou v proměnné offset.

Funkce MirrorY: Funkce zkopíruje všechny vzorky vstupního signálu do výstupního včetně metadat. Pozice vzorků v souřadnicích XY jsou zrcadleny dle osy Y. Všechny X hodnoty vzorků jsou při kopírování nahrazeny hodnotou mínus X.

Funkce MirrorX: Funkce zkopíruje všechny vzorky vstupního signálu do výstupního včetně metadat. Pozice vzorků v souřadnicích XY jsou zrcadleny dle osy X. Interně je využívána funkce Scale s parametrem scaleFactor = -1.

Funkce ExtractYvalues: Funkce kopíruje Y hodnotu každého vzorku vstupního signálu do výstupního listu číselných hodnot. Výstupní datový typ je List<double>.

5.2.3 Třída Interpolation

Třída Interpolation obsahuje pomocné funkce pro převzorkování signálu využití polynomiálních interpolací.

Funkce SplitToContinuousParts: Funkce zajistí rozdělení rekonstruovaného signálu do několika částí tak, aby se v těchto částech nevyskytovaly skokové změny hodnoty větší, než je parametr FiltrEpsilon u rekonstruovaného signálu. Aby tyto části na sebe navazovaly, také tato funkce doplňuje vzorky do rekonstruovaného signálu tak, že v případě detekce skokové změny je doplněn vzorek, který by byl uložen jako poslední před skokovou změnou a jeho Y hodnota je rovna předchozímu vzorku. Tato funkce realizuje požadavky na úpravu signálu před vlastní interpolací definované v kapitole 2.2.1 .

Funkce SplitToSmoothParts: Funkce rozděluje rekonstruovaný signál na více částí v místě prudkých změn strmosti rekonstruovaného signálu. Tento krok zabraňuje vzniku nežádoucích oscilací Lagrangeova interpolačního polynomu. Při použití lineární interpolace se tato funkce nevyužívá.

Funkce OversampleSignal: Funkce zajišťuje převzorkování signálu s novou vzorkovací zadanou parametrem samplePeriod za použití instance vybraného typu, který implementuje rozhraní IPolynomialInterpolator.

5.2.4 Rozhraní iPolynomialInterpolator

Rozhraní iPolynomialInterpolator je společný předek všech dalších tříd pro výpočet interpolačních funkcí. Definuje funkce pro manipulaci s krátkým polem vstupních vzorků (uzlů interpolace) a funkce pro výpočet interpolace. Díky tomuto společnému rozhraní může být se všemi interpolačními funkcemi pracováno stejně.

5.2.5 Třída LinearInterpolator

V této třídě jsou implementovány funkce pro výpočet lineární interpolace. Pole uzlů interpolace má vždy 2 prvky. Koeficienty pro výpočet interpolační přímky jsou vypočítány vždy při změně prvků v poli uzlových bodů.

5.2.6 Třída LagrangeInterpolator

V této třídě jsou implementovány funkce, které realizují interpolaci N uzlových bodu pomocí Lagrangeova polynomu stupně N-1 . Algoritmus pro výpočet Lagrangeovy interpolace má složitost N^2 kde N je počet uzlových bodů. Také při interpolaci polynomu vyššího stupně dochází častěji k oscilacím interpolační funkce, nebo k jiným chybám ve tvaru funkce. Proto se využívá Lagrangeův interpolační polynom 3. stupně. Průběh je interpolován po částech, série uzlových bodů $[x_i, y_i]$ až $[x_{i+3}, y_{i+3}]$ je použita pro interpolaci x v intervalu $<x_{i+1},x_{i+2}>$. Pro interpolaci $x > x_{i+2}$ je zahozen uzlový bod $[x_i, y_i]$, zbylé body jsou posunuty o jednu pozici a je přidán nový uzlový bod $[x_{i+4}, y_{i+4}]$. Koeficienty Lagrangeova polynomu jsou přepočítány pro nové uzlové body a je vypočítána interpolace x v dalším intervalu. Výjimku tvoří počátek a konec signálu kde je počáteční čtveřice uzlových bodů $[x_0, y_0]$ až $[x_3, y_3]$použita pro interpolaci x v intervalu $<x_0,x_2>$ a konečná čtveřice bodů $[x_{n-3}, y_{n-3}]$ až $[x_n, y_n]$ je použita pro interpolaci x v intervalu $<x_{n-2},x_n>$. Matematicky je výpočet Lagrangeovy interpolace popsán v kapitole 3.1.3

V případech, kdy je v interpolovaném signálu prudká změna popsaná nízkým počtem uzlových bodů, průběh Lagrangeova interpolační polynomu se dostává mimo meze, definované

předchozím uzlovým bodem a nastavením vzorkovače. V těchto případech je odchylka interpolace od předchozího uzlového bodu limitována dle nastavení vzorkovače. Jak oscilace interpolační funkce, tak samotné oříznutí, rozšiřuje spektrum signálu o nové složky. Tato skutečnost omezuje praktické použití Lagrangeovy interpolace.

Při výpočtu Lagrangeovy interpolace je použito několik funkcí, které se opakují v cyklu pro výpočet každého bodu interpolační funkce. Tyto funkce obsahují pouze několik matematických operací a proto nezanedbatelná část času potřebná k vykonání této funkce je spotřebována na volání funkce a návrat zpět do místa odkud byla volána. U těchto funkcí bylo otestováno použití atributu funkce [MethodImpl(MethodImplOptions.AggressiveInlining)], který informuje překladač, že tato kód této funkce má být vložen na místo odkud je volána (Inlining). Výhodou je, že program zůstává díky funkci přehledný, a stejná část kódu nemusí být v programu vypsána několikrát, jako v případě, kdyby místo volání funkce byl v tomto místě umístěn přímo její zdrojový kód. Výsledný program je zároveň vykonáván rychleji, protože nedochází k volání funkcí a návratu. Dle testů přinesla tato úprava zlepšení výkonu o 5 až 10% v závislosti na počtu bodů, ve kterých byla interpolace počítána. Při zvyšujícím se počtu interpolovaných bodů mezi uzly interpolace bylo procentuální zlepšení výraznější.

5.2.7 Třída Integration

Ve statické třídě Integration jsou umístěny funkce pro numerické integrování. V této třídě jsou obsaženy statické metody, jejichž výstupem je číslo reprezentující hodnotu určitého integrálu, nebo objekt datového typu Signal, reprezentující integrální funkci. Pro výpočet sumy jednotlivých části integrálu je z důvodu přesnosti u všech funkcí v této třídě použit Kahanův algoritmus kompenzovaného součtu implementovaný v třídě DoubleMath.Summator.

Funkce TrapezoidalIntegration: Funkce pro výpočet přibližné integrální funkce vstupního signálu pomocí lichoběžníkového pravidla. Lichoběžníkové pravidlo podrobněji popsáno v kapitole 3.2.1

Funkce RectangleAproximation: Funkce pro výpočet přibližné integrální funkce vstupního signálu pomocí obdélníkového pravidla. Obdélníkové pravidlo podrobněji popsáno v kapitole 3.2.2. Také je možno určit, zda se použije pravidlo levého nebo pravého koncového bodu. Pokud tento parametr není vyplněn, jako výchozí je používáno pravidlo levého koncového bodu.

Funkce DefiniteIntegral: Tato funkce zajišťuje výpočet určitého integrálu (plochy pod křivkou) zadaného vstupního signálu, nebo jeho části. V případě definice požadavku na výpočet určitého integrálu pouze z části signálu, je interně použita funkce Signal.SubSignal() pro výběr požadované části. Při výpočtu určitého integrálu není ukládán průběh integrální funkce, proto je výpočet rychlejší a méně náročný na paměť, než výpočet integrální funkce a přečtení jejího posledního vzorku. Tato funkce využívá pro integraci Lichoběžníkové pravidlo, které dle testů obvykle poskytuje menší odchylku.

5.2.8 Třída Derivation

Třída Derivation obsahuje metody pro přibližný numerický výpočet numerické derivační funkce.

Funkce SimpleDerivation: Funkce pro výpočet derivace dle matematického popisu v kapitole 3.3.

Funkce FiltredDerivation: Funkce implementující robustnější výpočet derivace, který podává lepší výsledky, pokud je vstupní signál zatížen šumem. Matematicky popsáno v kapitole 3.3.1. Pro správnou funkci je vyžadováno, aby vstupní signál byl vzorkován ekvidistantně.

5.2.9 Třída Statistics

Třída obsahující základní statistické funkce. Většina metod této třídy byla vytvořena jako pomocná pro výpočet korelačního koeficientu. Mohou však najít využití i jinde, a proto jsou dostupné z vnějšího rozhraní knihovny.

Funkce AverageValue: Vypočítá aritmetický průměr všech hodnot vzorků vstupního signálu. Pro výpočet průměru je použita třída AvgCounter, která pro výpočet sumy využívá Kahanův sumační algoritmus.

Funkce Variance: Vypočítá rozptyl Y hodnot vzorků vstupního signálu. Je využit AvgCounter pro výpočet průměru kvadratické odchylky od střední hodnoty.

Funkce StandartDeviation: Vypočítá směrodatnou odchylku statistického souboru všech Y hodnot vzorků vstupního signálu. Interně využívá funkci Variance.

Funkce Covariance: Z hodnot vzorků dvou vstupních signálu vypočítá kovarianci. Vstupní signály musí mít stejný počet vzorků. Střední hodnoty vstupních posloupností jsou počítány pouze jednou. Časová náročnost algoritmu je lineárně závislá na počtu vzorků.

Funkce CorelationCoeficient: Vypočítá Pearsonův párový korelační koeficient dle popisu v kapitole 3.4.1. Interně využívá funkce Covariance a StandartDeviation. Vstupní signály musí mít stejný počet vzorků.

5.2.10 Třída FFT

Funkce ComplexFFT: Provádí výpočet rychlé Fourierovy transformace nad polem komplexních čísel, výstupem transformace je opět pole komplexních čísel. Vzhledem k použitému algoritmu Radix 2 DIT musí být délka vstupního pole 2^N. Dle velikosti vstupního pole je určen počet stupňů, ve kterých bude algoritmus FFT probíhat. Nezbytným krokem před výpočtem FFT pomocí Radix 2 DIT je přeuspořádání vstupních dat do bitově reverzního pořadí. Bohužel architektura x86 neumožňuje rychlý výpočet bitově reverzního pořadí, neboť procesory nejsou vybaveny potřebnou instrukcí a změnu pořadí bitů je možno provést pomocí cyklu bitových posunů a základních logických operací ovšem časová náročnost této úpravy vysoká. Proto byla pro potřebu bitové reverzace vytvořena třída BitReversalTool. Tato třída obsahuje vyhledávací tabulku (lookup table) o velikosti 256 prvků která pro všechny hodnoty bajtu dokáže rychle vrátit hodnotu v bitově reverzním pořadí. Hodnota indexu datového typu Int32 je rozložena na jednotlivé bajty, ty jsou pomocí vyhledávací tabulky převedeny do bitově reverzního pořadí a výsledek je posunut o tolik bitů, aby zůstalo zachováno pouze N bitů nezbytných k určení adresy v poli o velikosti 2^N. Časová náročnost cyklu je lineárně závislá na počtu vzorků, přeuspořádání probíhá “Out of place” tzn. data jsou kopírována na jiné místo v paměti. Nevýhodou je vyšší paměťová náročnost, ale v budoucnu bude možné snadno algoritmus upravit pro paralelní zpracování pomocí více procesorových jader.

Vlastní výpočet FFT probíhá dle postupu popsaného v kapitole 3.6.1. Vlastní algoritmus není napsán rekurzivně ale pomocí vnořených cyklů. Tento způsob výpočtu značně zjednodušuje použití tabulky pro předpočítání twidle faktoru. Výpočet twidle faktoru je časově náročnou komplikací protože obsahuje výpočty goniometrických funkcí sinus a kosinus. Stejné hodnoty twidle faktoru se při výpočtu FFT v jednotlivých stupních mnohokrát opakují, proto je výhodné provést výpočet twidle faktoru pro každou hodnotu parametru pouze jednou a výsledek uložit do tabulky pro pozdější opakované použití. Twidle faktor je po této úpravě počítán pouze N/2 krát namísto $(N/2).Log_2(N)$ krát. Tato úprava dle testů přinesla zrychlení výpočtu FFT přibližně o 30%.

Funkce SignalFFT: Převádí Y hodnoty vzorků vstupního signálu na pole objektů typu Complex. Pokud počet vzorků vstupního signálu není roven 2^N kde N je celé číslo, je doplněno tolik nových vzorků s Y hodnou 0, aby výsledný počet vzorků 2^N kde N je nejnižší možné celé číslo. Nad tímto polem je proveden výpočet FFT. Výstupní data jsou vložena do nově vytvořené instance třídy Signal. Z výstupu funkce ComplexFFT je kopírováno do nové instance třídy Signal pouze N/2 vzorků. Toto zjednodušení je možné díky tomu, že pro vstupní posloupnost, která obsahuje pouze reálná čísla, je výsledek FFT vždy symetrický.

5.2.11 Třída CrossCorelation

Tato třída obsahuje metody pro výpočet dynamických parametrů korelační analýzy. Jsou implementovány metody pro výpočet korelační a autokorelační funkce. Matematicky popsáno v kapitole 3.4.2.

Funkce SignalCrossCorelation: Tato funkce realizuje výpočet vzájemné korelační funkce dvou signálů. Oba vstupní signály musí být ekvidistantně vzorkovány a musí mýt stejnou periodu vzorkování. Výpočet vzájemně korelační funkce je proveden pro posunutí z intervalu

$<-N_B;N_A+N_B>$ kde N_A je počet vzorků prvního vstupního signálu a N_B je počet vzorků druhého vstupního signálu.

Funkce Autocorelation: Interně využívá funkci SignalCrossCorelation. Funkce má jeden parametr, který je do funkce SignalCrossCorelation předáván na místě obou vstupních parametrů.

5.2.12 Třída Convolution

Tato třída obsahuje metodu pro výpočet lineární konvoluční funkce. V budoucnu by zde mohla být snadno doplněna metoda pro výpočet kruhové konvoluční funkce.

Funkce LinearConvolution: Výpočet lineární konvoluční funkce lze po úpravě vstupních dat převést na výpočet vzájemné korelační funkce, popsáno v kapitole 3.5. Vstupní signál definovaný parametrem b je časově převrácen funkcí MirrorY a pak spolu s neupraveným signálem, který je definován parametrem a, vstupují do funkce SignalCrossCorelation.

5.2.13 Třída Summator

V této třídě je implementován Kahanův sumační algoritmus kompenzovaného součtu nezbytný pro přesnější výpočet sumy z velkého množství prvků [14]. Princip činnosti tohoto algoritmu je možno popsat na následující ukázce zdrojového kódu. Proměnné `sum` a `correction` jsou externí proměnné datového typu double, které mezi voláním následující funkce nemění svoji hodnotu. Výsledek kumulovaného součtu je k dispozici v proměnné `sum`.

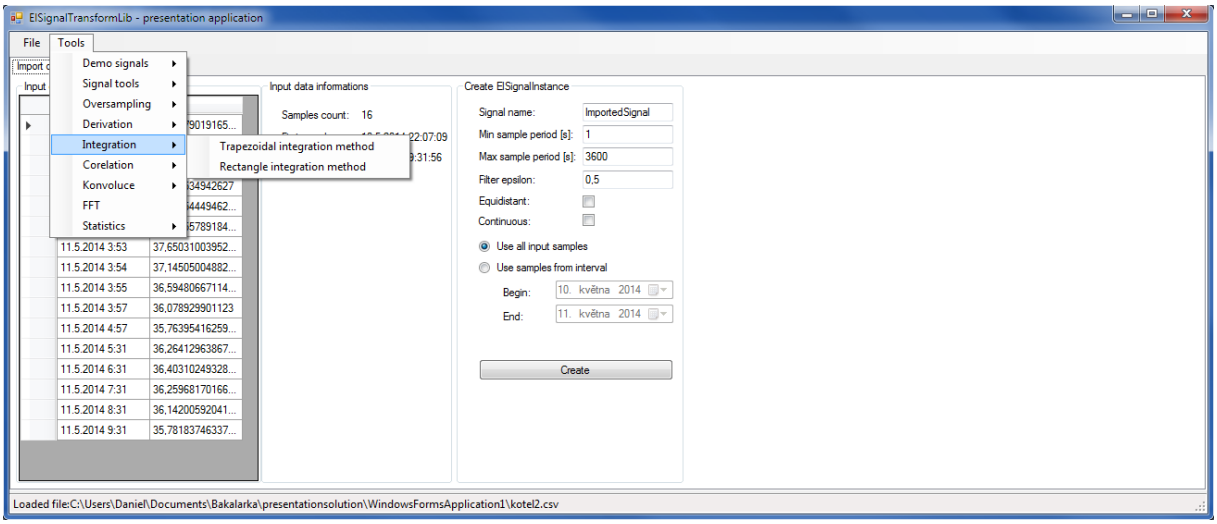
```
void Add(double value) {                //funkce pro přičtení čísla k sumě
       double y = value - correction;   //úprava   přičítané   hodnoty   o   korekci
                                        //z předchozí iterace
       double t = sum + y;              //pokud je y o mnoho řádu vyšší než sum dojde
                                        //k zaokrouhlovací chybě
       correction = (t - sum) - y;      //vypočet nové hodnoty korekce
       sum = t;                         //aktualizace sumy
}
```

Algoritmus přičtení hodnoty k již akumulované sumě se skládá z několika po sobě jdoucích sčítacích operací. Z toho vyplývá, že tento krok není na rozdíl od běžného sčítání atomický - tzn. průběh vykonávání procedury sčítání může být přerušen zpracováním jiného programového vlákna. Toto omezuje možnost použití pouze na jedno-vláknové úlohy. Pro použití ve více

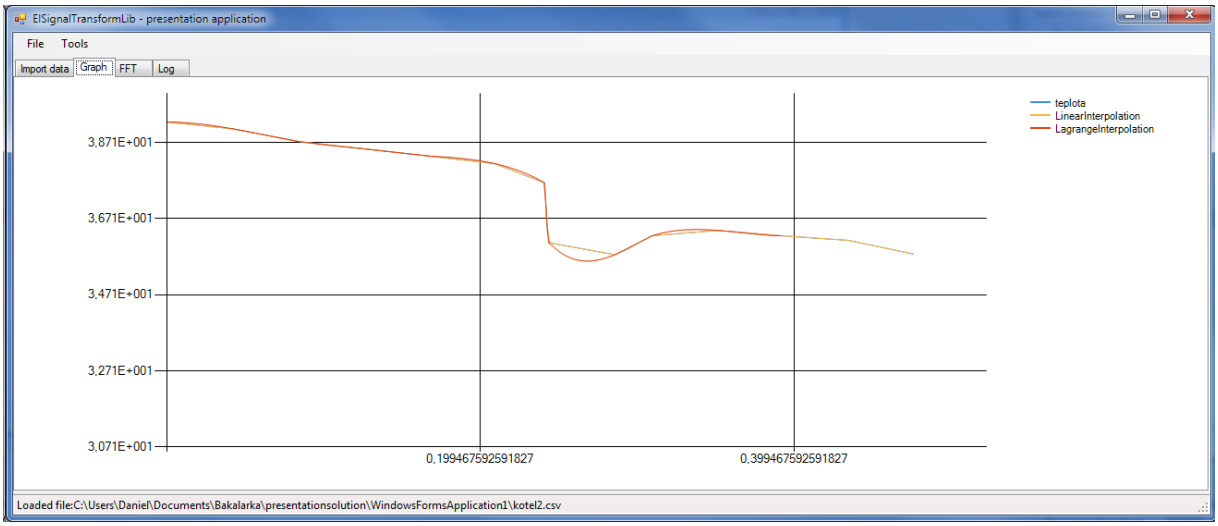
vláknovém prostředí by bylo nezbytné umístit funkci přičtení hodnoty k sumě do kritické sekce. Kontrola výlučného přístupu do kritické sekce je značně náročná na procesorový čas. Proto by v případě použití ve více vláknovém prostředí bylo vhodné, aby jedna instance třídy Summator byla využívána pouze jedním vláknem.

5.3 Ukázková aplikace

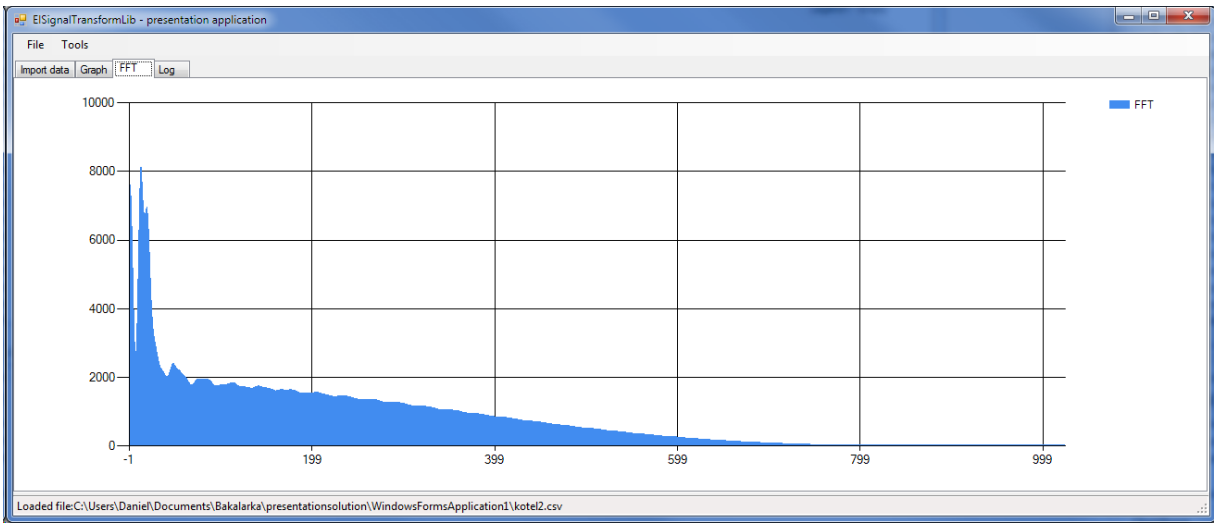
Pro testovací účely a ukázku použití funkcí v knihovně ElSignalTransformLib byla vytvořena jednoduchá formulářová aplikace, která umožňuje načtení dat z .csv souboru, vytvoření instance třídy Signal do které jsou vloženy vzorky ze souboru. Mohou být vloženy všechny vzorky nebo jejich část definovaná časovým intervalem uložení vzorku. Dále také ukázková aplikace umožňuje generování několika předdefinovaných typů signálů. Výsledky jsou zobrazeny v grafu, pokud jde o funkci. V případě, že je výsledkem číslo, je zobrazeno v dialogovém okně. Při výběru požadované funkce z menu "Tools" je zobrazeno dialogové okno pro zadání parametrů volané funkce, jako je signál který má být zpracován, a v případě potřeby i další konstanty.



Obrázek 18 - prezentační aplikace - záložka importu dat.



Obrázek 19 - prezentační aplikace - záložka grafu.

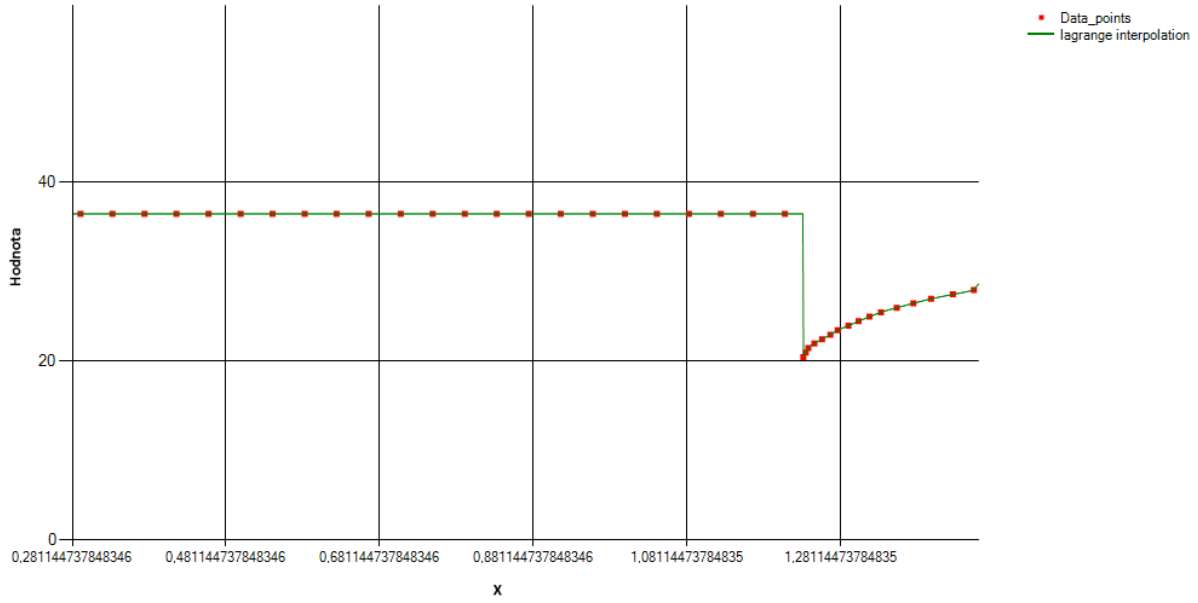


Obrázek 20 - prezentační aplikace - zobrazení spektra signálu

6 Závěr

6.1 Dosažené výsledky

Algoritmus pro rozdělení signálu do více spojitých částí se ukázal, jako funkční. Na základě informací o nastavení vzorkovače dokáže doplnit interpolační body tak, aby mohly být realisticky interpolovány úseky signálu, jako je například zapnutí systému, či náhlé změny v měřené soustavě.



Obrázek 21 - Rozdělení signálu do spojitých části a interpolace po částech

Implementace lineární interpolace byla snadná. Výstup lineární interpolace odpovídá teoretickým předpokladům. Nevýhodou lineární interpolace je nespojitá první derivace. Výhodou lineární interpolace je rychlost výpočtu. Rychlost byla testována na vzorku dat, který obsahoval 9422 uzlových bodů, interpolace byla vypočítána v 193375 bodech za 107 ms.

Implementace Lagrangeovým polynomem 3. řádu na testovacích datech pracovala velice dobře. Při testech na reálných datech se objevil problém v podobě nežádoucích oscilací interpolační funkce v okolí prudké změny strmosti interpolované funkce. Tento problém je řešitelný rozdělením průběhu do více částí interpolovaných nezávisle na sobě. Problém nastává, pokud jsou náhlé změny popsány nízkým počtem vzorků. Na dosud pořízených datech, která byla uložena s nastavením vzorkovače, které preferovalo požadavky na malé množství uložených dat před přesným zachycením průběhu signálu, nelze interpolaci Lagrangeovým polynomem považovat za 100% spolehlivou. Nasazení v praxi bude možné pouze tehdy, pokud tomu bude přizpůsobeno nastavení vzorkovače.

Výkonový test výpočtu interpolace pomocí Lagrangeova polynomu byl proveden na stejných datech jako test lineární interpolace. Počet uzlových bodů 9422, interpolace vypočítána v 193375 bodech za 367ms.

Implementace funkcí pro numerickou integraci, derivaci a korelační analýzu, byla z programátorského hlediska snadná. Pro korekci zaokrouhlovacích chyb při integraci byl implementován algoritmus kompenzovaného součtu. Funkce pro výpočet rychlé Fourierovy transformace byla složitější zejména z důvodu výkonové optimalizace bitově reverzního řazení.

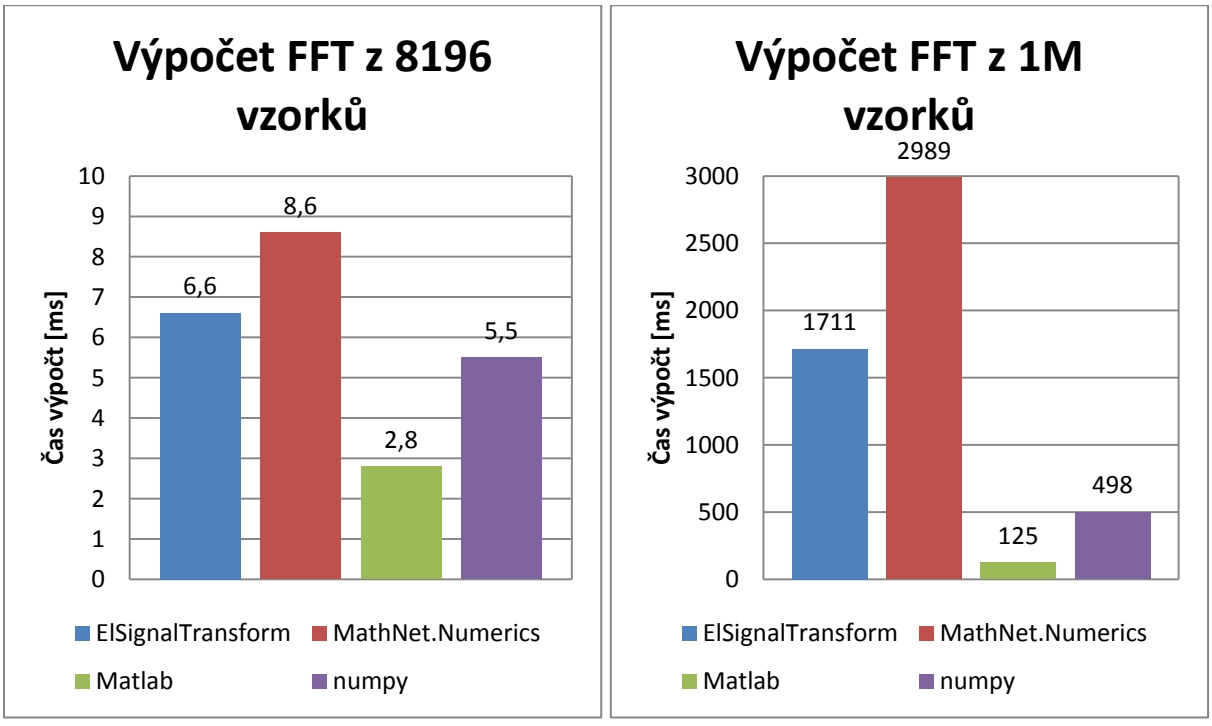
6.2 Srovnání s jinými softwarovými knihovnami

Funkce pro rekonstrukci signálu, která splňuje požadavky definované specifickým způsobem jeho vzorkování je poměrně unikátní. Ostatní funkce knihovny implementují běžné numerické metody zpracování signálu, které je možno nalézt i v jiných knihovnách. Například knihovna numpy pro programovací jazyk python nebo MathNet.Numerics pro C#. Knihovna numpy obsahuje téměř všechny funkce jako mnou vytvořená knihovna. Numpy také obsahuje mnohé běžné matematické funkce, které jsou v C# implementovány systémovou knihovnou Systém.Math. Výhodou knihovny numpy jsou pokročilé výkonové optimalizace pro moderní procesory s využitím SIMD instrukcí (Single Instruction Multiple Data). Nevýhodou je komplikované spuštění python programu z prostředí asp.net.

Knihovna MathNet.Numerics obsahuje nejen všechny funkce mnou vyvinuté knihovny, ale také mnohé další matematické funkce, práci s maticemi, numerické řešení soustav rovnic a podobně. Kód této knihovny je napsán přehledně a její použití je možné zdarma. Bohužel projekt Math.NET byl nalezen až po dokončení velké části práce na mojí knihovně.

Výkonové srovnání FFT

Vstupní data – obdélníkový signál s periodou 64 vzorků. Délka testovaného souboru 8196 a 1 milion vzorků. U testu doby výpočtu FFT z 8196 vzorků z důvodu značného rozptylu výsledků byl test opakován 5x pro každé řešení a odebrány značně odchýlené hodnoty do srovnání zařazeny průměrné hodnoty.



Všechny výkonové testy probíhaly na počítači s procesorem Intel Core i5 na frekvenci 2,67Ghz. Program využívá pouze jedno procesorové jádro.

6.3 Plány pro budoucí rozvoj

- Integrace do informačního systému MASA
- Podpora pro více vláknové zpracování.
- Rozšíření Statistics, např. o lineární regresi

7 Použité zdroje informací

Citované zdroje

[1] FAJMON, Břetislav a Irena RŮŽIČKOVÁ. 2003. *Matematika 3* [online]. Dostupné z: www.umat.feec.vutbr.cz/~fajmon/bma3. Elektronický text UMAT FEKT.

[2] *Integration: Left, Right and Trapezoid Rules* [online]. Dostupné z: https://www.math.ohiou.edu/courses/math3600/lecture21.pdf

[3] HLAVÁČ, Václav a Milan SEDLÁČEK. *Zpracování signálů a obrazů.* Skriptum FEL ČVUT Praha, 2002

[4] ŠEBESTA, Vladimír a Zdeněk SMÉKAL. *Signály a soustavy* [online]. 2004 [cit. 2015-05].

[5] MELOUN, Milan. *Kompendium statistického zpracování dat: metody a řešené úlohy*. Vyd. 2., přeprac. a rozš. Praha: Academia, 2006, 982 s. ISBN 80-200-1396-2.

[6] RAŠEK, Petr a Jakub Šťastný. *FFT motýlek* [online] Praha: ČVUT FEL Katedra Teorie obvodů, 2006. Výzkumná zpráva #Z06-6. Dostupné z: http://amber.feld.cvut.cz/fpga/studenti/petr_rasek/fft_butterfly.pdf

[7] *Rychlá Fourierova transformace (FFT) pro AVR* [online]. [cit. 2015-05-23]. Dostupné z: http://elektronika.kvalitne.cz/ATMEL/necoteorie/transformation/AVRFFT/AVRFFT.html

[8] *FFT in C* [online]. [cit. 2015-05-23]. Dostupné z: http://www.katjaas.nl/FFTimplement/FFTimplement.html

[9] HIGHAM, Nicholas J. *Accuracy and stability of numerical algorithms*. 2nd ed. Philadelphia: Society for Industrial and Applied Mathematics, c2002, xxx, 680 p. ISBN 0898715210.

[10] C# language reference. [online]. [cit. 2014-12-18]. Dostupné z: http://msdn.microsoft.com/en-us/library/618ayhy6.aspx

[11] QUARTERONI, Alfio a Fausto SALERI. *Scientific computing with MATLAB*. New York: Springer, c2003, ix, 257 p. ISBN 35-404-4363-0.

[12] SEAN ERON ANDERSON,. *Bit Twiddling Hacks* [online]. [cit. 2015-05-23]. Dostupné z: http://graphics.stanford.edu/~seander/bithacks.html

[13] Microprocessor Standards Committee of the IEEE Computer Society. *IEEE standard for floating-point arithmetic*. New York, NY: Institute of Electrical and Electronics Engineers, 2008. ISBN 9780738157528.

[14] KAHAN, W. Pracniques: further remarks on reducing truncation errors. *Communications of the ACM* [online]. 1965, vol. 8, issue 1 [cit. 2015-05-26]. DOI: 10.1145/363707.363723.

[15] WEISSTEIN, Eric W. *Covariance* [online]. [cit. 2015-05-26]. Dostupné z: http://mathworld.wolfram.com/Covariance.html

Různé další zdroje

- NEVŘIVA, Pavel. *Analýza signálů a soustav: metody a řešené úlohy*. 1. vyd. Praha: BEN - technická literatura, 2000, 671 s. ISBN 80-730-0004-0.
- NET framework reference source. [online]. [cit. 2014-12-18]. Dostupné z: http://referencesource.microsoft.com/
- JAN, Jiří. *Číslicová filtrace, analýza a restaurace signálů*. 2. upr. a rozš. vyd. Brno: VUTIUM, 2002, 427 s. ISBN 80-214-1558-4.
- http://numerics.mathdotnet.com – domovská stránka projektu Math.NET numerics
- http://www.numpy.org/ - domovská stránka projektu numpy